DYNAMIC INTERACTIONS FOR NETWORK VISUALIZATION AND SIMULATION

THESIS

Çiğdem Yetişti, First Lieutenant, TUAF

AFIT/GE/ENG/09-50

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GE/ENG/09-50

DYNAMIC INTERACTIONS FOR NETWORK VISUALIZATION AND SIMULATION

THESIS

Presented to the Faculty
Department of Electrical and Computer Engineering
Graduate School of Engineering and Management
Air Force Institute of Technology
Air University
Air Education and Training Command
In Partial Fulfillment of the Requirements for the
Degree of Master of Science in Electrical Engineering

Çiğdem Yetişti, B.S.E.E.
First Lieutenant, TUAF

March 2009

AFIT/GE/ENG/09-50

DYNAMIC INTERACTIONS FOR NETWORK VISUALIZATION AND SIMULATION

Çiğdem Yetişti, B.S.E.E.
First Lieutenant, TUAF

Approved:

/signed/	05 Mar 2009
Lt Col Stuart Kurkowski, PhD(Chairman)	date
/signed/	05 Mar 2009
Dr. Kenneth M. Hopkinson (Member)	date
/signed/	05 Mar 2009
Capt Ryan W. Thomas, PhD (Member)	date

AFIT/GE/ENG/09-50

Abstract

Most network visualization suites do not interact with a simulator, as it executes. Nor do they provide an effective user interface that includes multiple visualization functions. The subject of this research is to improve the network visualization presented in the previous research [5] adding these capabilities to the framework. The previous network visualization did not have the capability of altering specific visualization characteristics, especially when detailed observations needed to be made for a small part of a large network. Searching for a network event in this topology might cause large delays leading to lower quality user interface. In addition to shortfalls in handling complex network events, [5] did not provide dynamic user interactions since it did not have real-time interaction with a simulator. These shortfalls motivate the development of a new network visualization framework design that provides a more robust user interface, network observation tools and an interaction with the simulator. Our research presents the design, development and implementation of this new network visualization framework to enhance network scenarios and provide interaction with NS-2, as it executes. From the interface design perspective, this research presents a prototype design to ease the implementation process of the framework. The visualization functions such as clustering, filtering, labeling and color coding help accessing network objects and events, supporting four tabs consisting of buttons, menus, and sliders. The new network visualization framework design gives the ability to handle the inherent complexity of large networks, allowing the user to interact with the current display of the framework, alter visualization parameters and control the network through the visualization. In our application, multiple visualizations are linked to NS-2 to build execution scenarios which let to test clustering, filtering, labeling functionalities on separate visualization screens, as NS-2 progresses.

Acknowledgements

I'd like to thank to my thesis advisor, Lt Col Stuart H. Kurkowski, for his patience, advice and help through the project. Also to his wife, for her help and encouragement for my presentation. I would like to thank Josh Abernathy from Cedarville college for his contributions towards building the new network visualization framework and his extensive knowledge of *prefuse*. I am ever thankful to my family, who supported me and kept me believing all things were possible. Without your faith in me, I don't know if I would have had the strength to complete this process. Lastly, I'd like thank to my sponsor family for providing a home atmosphere, being my family away from home and giving me faith when I needed it.

Çiğdem Yetişti

Table of Contents

Page

Page

List of Figures

Figure Page

Figure Page

Figure Page

Figure Page

List of Abbreviations

Abbreviation		Page
JSE	Joint Service Environment	3
NetViz	Network Event Visualization	4
NS-2	The Network Simulator 2	4
VINT	Virtual Inter Network Test-bed	4
OTcl	Object Oriented Tool Command Language	4
NAM	The Network Animator	5
Tcl	Tool Command Language	5
UAV	Unmanned Aerial Vehicle	7
GUI	Graphical User Interface	8
MVC	Model-View-Controller	8
nans	Network Analyzer for Network Simulator	12
UI	The User interface	14
VISUAL	Visual Information Security Utility for Administration Live	16
MBGP	Multicast Border Gateway Protocol	20
EID	Ecological interface design	22
NNM	Network Node Manager	22
VLANs	Virtual Local Area Networks	23
JSE	Joint Service Environment	49
RTT	Round Trip Time	52
CCT	Configurable Command Tool	58

DYNAMIC INTERACTIONS
FOR NETWORK VISUALIZATION AND SIMULATION

I. Introduction

Today's network development requires high performance and, functionality along with tools that can display transmissions and analyze them. To understand the structure of large networks, network simulators can be used as simulation environments in order to generate the events for the real-world network scenarios. Network simulators are relatively fast and inexpensive when compared to the cost and time involved in setting up an entire testbed containing multiple networked computers, routers and data links. Running a simulator produces a block of data including keywords or numbers showing the scenario results. A detailed performance for these results is an important contribution to the evaluating of the object relationships, data structures, control flow and data flow. Instead of the raw data, visual representations are highly desirable to obtain this analysis. The reason is that the visual perception of humans allows much faster interpretation of structural and geometric information than any other kind of information [1]. For this reason, network visualization suites are widely utilized to display network events.

But in some cases, the complexity and scale of the network makes it nearly impossible to determine the needed part of the network topology to display on the visual presentation. Searching for a network event in this topology might cause large delays leading to lower quality user interface. Multiple network visualization functions can be used to improve the user interfaces and make the large volume of data comprehendible by human users. The subject of this research is building a framework, which facilitates these visualization functions to assist the investigation of network objects and events.

Simulators can have their visualization capabilities integrated or as additional software suites. Visualization suites enhance the awareness of operators by providing goal-oriented user interfaces which support the way operators view their networks and their network operation activities. Supporting the visual presentation with these user interfaces including interactive functions gives the ability to dynamically change the graphical representations and receive response to user inputs.

The simulator utilized in this research, Network Simulator-2 (NS-2) [2], has a visualization package called Network Animator (NAM) [3, 4], which is discussed in Chapter II. In the previous study [5], a network visualization (NetViz) was also produced as a visualization suite which is similar to NAM. NetViz could visualize how the network performed during a specific scenario using a detailed trace file provided by NS-2. However, the NetViz framework is only capable of handling simple network events and has shortfalls in dynamic user interactions. It does not have real-time interaction with a simulator. This motivates the need for a new NetViz framework design that provides a more robust user interface, network observation tools and a real-time interaction with the simulator. Our research presents the design, development and implementation of this new NetViz framework design to enhance network scenarios and provide real-time interaction with NS-2. The new framework design gives the ability to handle the inherent complexity of large networks, allowing the user to interact with the current display of the framework on the fly and control the network through the visualization.

The interface design describes how the software communicates within itself, with systems that interoperate with it, and with humans who use it. From the interface design perspective, this research presents a prototype design employing researched studies discussed in Chapter II and design specifications described in Chapter III. Before constructing the prototype design, these specifications are developed using Heuristic Evaluation Guideline [6, 7] defined in Chapter II. The prototype design is introduced to attempt to better facilitate using multiple network visualization functions meeting user needs described in the design specifications. Providing a route to the development process of the new NetViz framework, this prototype design eases the implementation process. Each feature of this prototype design is described introducing possible contributions of these features to the new NetViz framework design.

A brief overview of toolkits, *prefuse* [8,9], *tracegraph* [10], *Network Analyzer for Network Simulator NS-2* (*nans*) [11] and *mediator* [12] is given in this research. The new framework design extends the widely used the *prefuse* visualization toolkit. This research extends many *prefuse* classes to customize network visualization and user preferences providing various dynamic interactions. Using *nans* and *tracegraph*, post processing tools, our research analyzes a simple wired scenario in detail to obtain 2D and 3D graphs for this network scenario. Usage of these post processing tools provides a useful foresight which would help the progress of the NetViz framework as examples graphs to plug into the user interface. These tools serve as guides to developing our application and future applications of graphs for NetViz. The new framework design also utilizes the *mediator* tool [12] which provides a connection between the NetViz and NS-2 by basically passing data and commands, bringing a kind of dynamism to the static simulation. The visualization functions [13] in the new NetViz framework such as clustering, labeling and color coding assist in accessing network objects and events. These functions support four tabs consisting of buttons, menus, and sliders which provide control over the network objects for different tasks.

This research tests the new NetViz framework functionality visualizing a complex the Joint Service Environment (JSE) scenario and a simple scenario. This research also builds execution scenarios to test clustering, filtering, labeling functionalities on separate visualization screens, linking multiple visualizations to NS-2 using *mediator*. The visual inspections of the framework are compared to the visualization outputs of NAM and the old NetViz framework.

II. Interactive Network Visualization and Related Studies

The previous research [5] by Captain Belue of USAF designed and developed a network event visualization (NetViz) framework using an open source visualization toolkit. In the following section, the network visualization features and components that are common to our research are overviewed as network environment since the new design of the framework requires a comprehensive understanding of these items. The other sections provide descriptions of toolkits used in this research, discussions of related network visualization studies and interactivity in the network visualization.

2.1 Network Environment

2.1.1 NS-2. The Network Simulator 2 (NS-2) was used as a simulation environment in order to generate the events for the NetViz [5]. NS-2 is the second version of a network simulator tool developed by the Virtual Inter Network Test-bed (VINT) project [2]. It is an event-driven network simulator, which is popular within the networking research community. NS-2 is an Object Oriented Tool Command Language (OTcl) [2] script interpreter which has a simulation event scheduler, network component object libraries and network setup (plumbing) module libraries (shown in Figure 2.1).

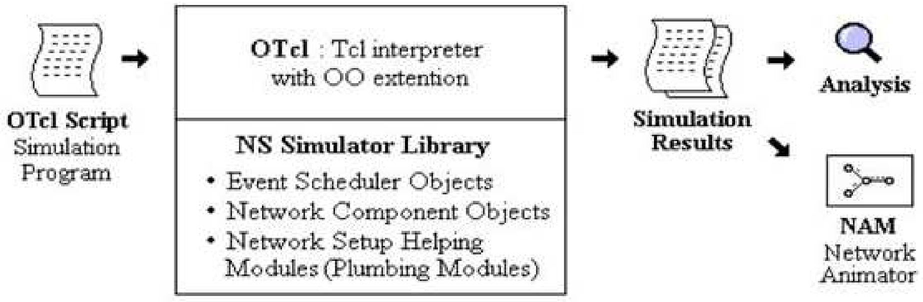

OTcl Script
Simulation Program
OTcl : Tcl interpreter with OO extention
NS Simulator Library
• Event Scheduler Objects
• Network Component Objects
• Network Setup Helping Modules (Plumbing Modules)
Simulation Results
Analysis
NAM
Network Animator

Figure 2.1: Simplified User's View of NS-2 Environment [2].

A user writes a simulation program in OTcl script language to utilize NS-2 for setting up and running a network simulation. The OTcl script starts an event scheduler, and sets up the network topology. It also establishes traffic sources and when to start and stop transmitting packets through the event scheduler [2] .

NS-2 includes The Network Animator (NAM), which provides packet-level animation and protocol-specific graphs for design and debugging of network protocols [2]. When NS-2 executes a network scenario, it produces simulation results in the form of a trace file, containing network events that occurred during the scenario.

2.1.2 Softwares used with NS-2.

2.1.2.1 NAM. NAM introduces a visual interpretation of the network events that occurred during the scenario, using the NS-2 trace file [3]. It can be executed directly from a Tool Command Language (Tcl) script [4]. NAM has a graphical user interface which has components such as play, fast forward, rewind, pause, and display speed controller as shown in Figure 2.2. After running scenarios in NS-2, users can see network events visualized in the NAM window.

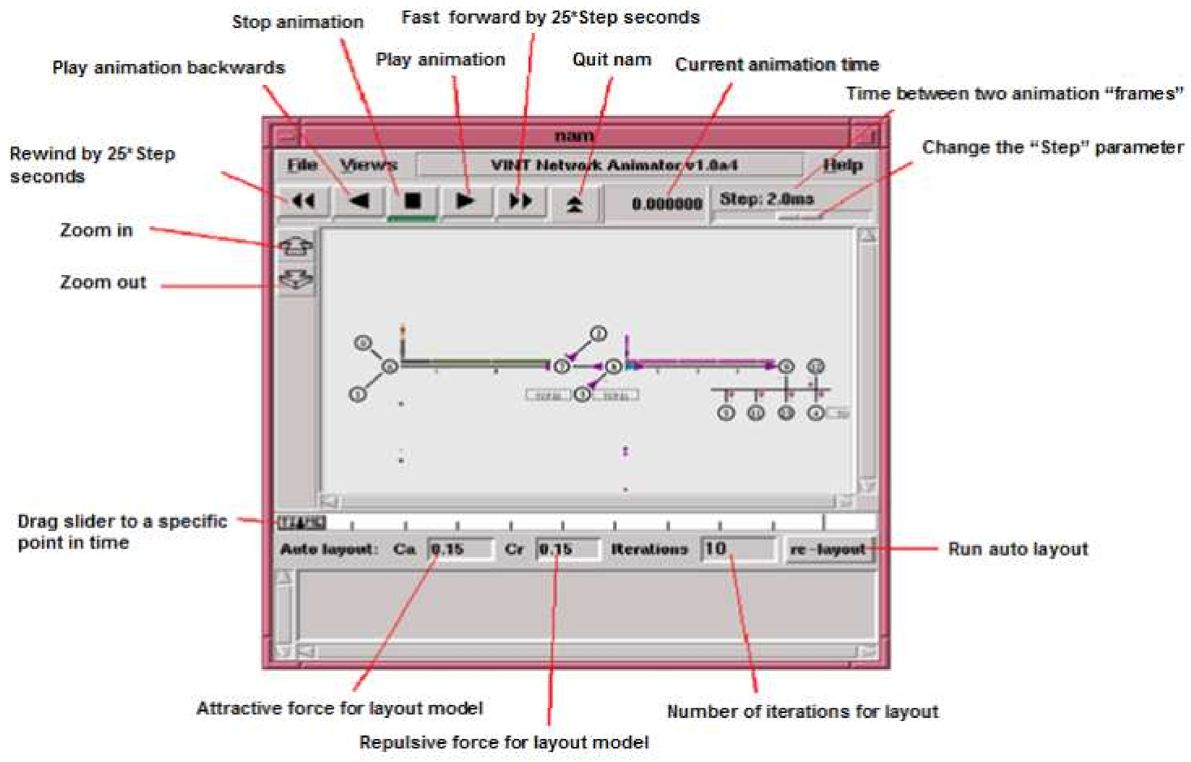
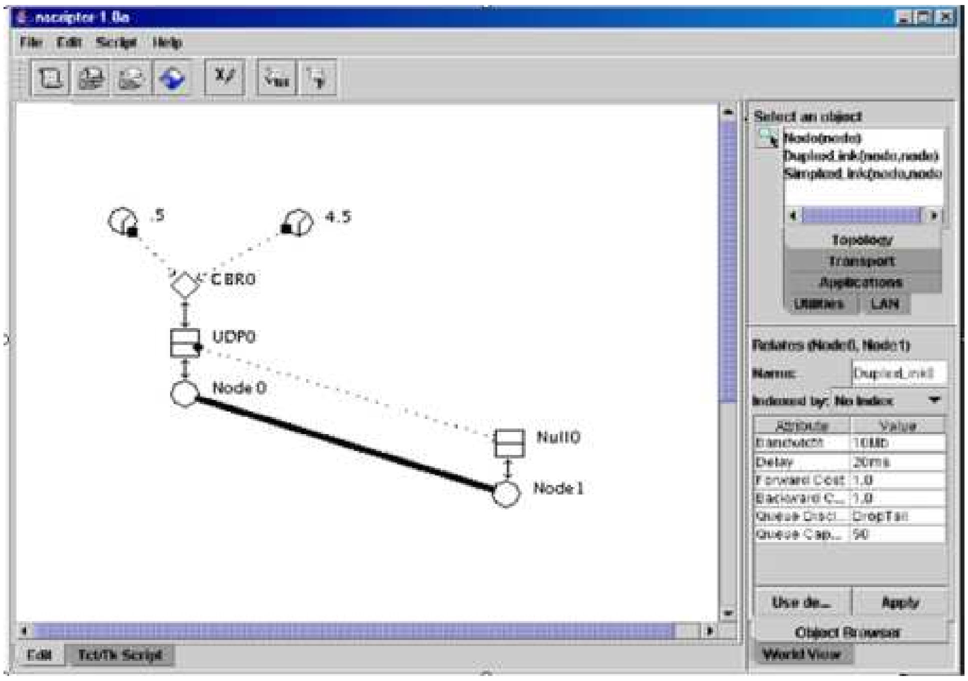

Stop animation
Fast forward by 25*Step seconds
Play animation backwards
Play animation
Quit nam
Current animation time
Time between two animation "frames"
Change the "Step" parameter
Rewind by 25*Step seconds
Zoom in
Zoom out
nam
File
Views
VINT Network Animator v1.0a4
Help
0.000000
Step: 2.0ms
Drag slider to a specific point in time
Auto layout:
Ca
0.15
Cr
0.15
Iterations
10
re-layout
Run auto layout
Attractive force for layout model
Repulsive force for layout model
Number of iterations for layout

Figure 2.2: A snapshot of Network Animator NAM showing packet traffic, queued packets, and dropped packets and NAM's graphical interface components [4].

For analysis information, NAM can graphically present information such as throughput and number of packets dropped at each link. It also provides a drag and drop interface for creating topologies.

2.1.2.2 Nscript. Nscript [14] is a graphical user interface for building NS-2 Tcl scripts. It is written in Java. The user builds topologies by adding nodes and links to the screen. The user also can create transport agents i.e. UDP, TCP, user defined libraries i.e. PING agent and, schedule simulation events i.e. sending/queuing packets. The scripts can be exported and run in NS-2 (see Figure 2.3).



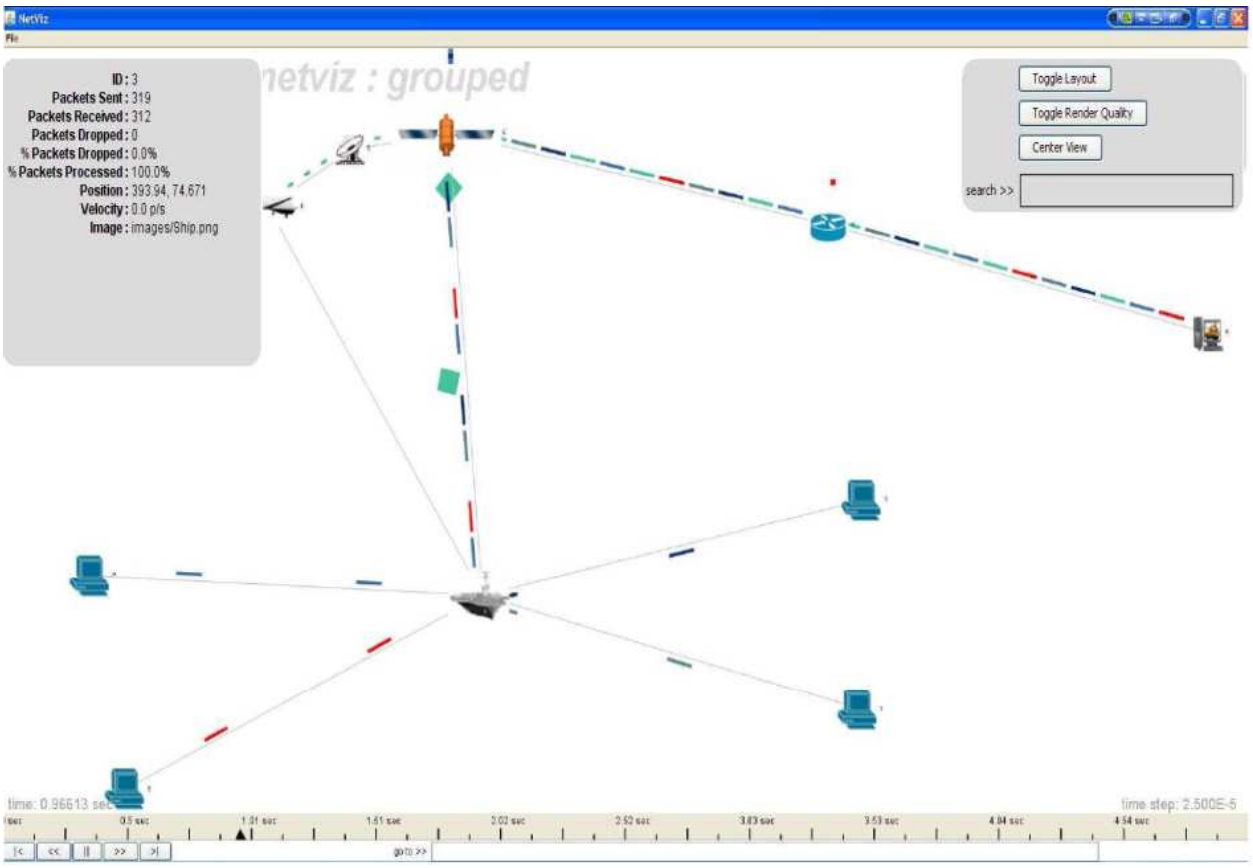
Relates (Node0, Node1)
Name: Duplexd_link0
Indexed by: No Index

Attribute	Value
Bandwidth	10Mb
Delay	20ms
Forward Cost	1.0
Backward C...	1.0
Queue Disci...	DropTail
Queue Cap...	50

Figure 2.3: Nscript application showing configuration of two nodes and the link for the simulation [49].

2.1.3 NetViz. Because of the shortages and shortfalls in the researched tools identified in [5], a network visualization framework was created as part of that research. The design, implementation, and testing of the NetViz was done to improve network visualization of both wired and wireless network events including: traditional packet animation across wired and wireless links, queue levels at each node, and dropped packets. *Prefuse* [8] toolkit was used to create the NetViz framework. Figure

2.4 shows a screenshot of NetViz visualizing a scenario for routers, a satellite, a ship and an Unmanned Aerial Vehicle (UAV) network.


ID : 3
Packets Sent : 319
Packets Received : 312
Packets Dropped : 0
% Packets Dropped : 0.0%
% Packets Processed : 100.0%
Position : 393.94, 74.671
Velocity : 0.0 p/s
Image : images/Ship.png

time: 0.96613 sec
time step: 2.500E-5

Figure 2.4: A screenshot of NetViz visualizing a scenario for routers, a satellite, a ship and an UAV communication [5].

Our research develops a new NetViz framework design that provides better tools for both observing network events and interacting with them. This new design is also developed utilizing the *prefuse* toolkit. More information about *prefuse* visualization toolkit will be given in the following section.

2.2 Toolkits Used in the Research

2.2.1 Prefuse Visualization Toolkit. *Prefuse* [8] is an extensible software framework that provides software developers with the ability to build their own interactive information visualization applications. *Prefuse* is written in the Java programming language with the help of Java 2D graphics library [9]. More than providing off the shelf widgets that serve as buttons or checkboxes in conventional Graphical User Interface (GUI) tools; *prefuse* provides a set of building blocks with higher granularity for constructing customized visualizations.

Prefuse can be utilized while creating separate applications, visual apparatus embedded in larger applications, and web applets. Comprising a library of design algorithms, navigation and interaction techniques, *prefuse* aims to significantly simplify the processes of representing and efficiently handling data, mapping data to visual representations, and building direct manipulation interaction with the visualized data [8].

The design of the *prefuse* toolkit is based upon a software architecture pattern which breaks up the visualization process into a series of separate steps, ranging from data acquisition and modeling to visual data encoding and presentation of interactive displays. This software architecture pattern is called the Information Visualization Reference Model and is depicted in the Figure 2.5.

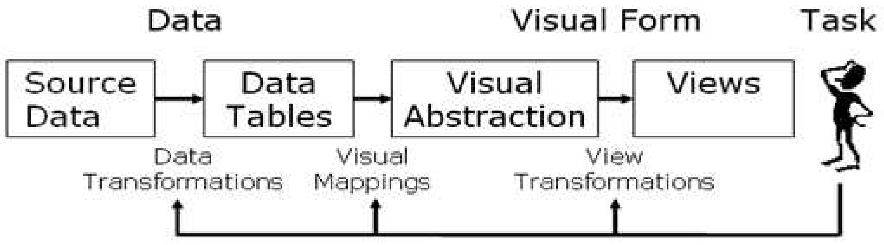


Figure 2.5: Diagram depicting the Information Visualization Reference Model of the Prefuse toolkit [15].

The *prefuse* toolkit is suitable for the Model-View-Controller (MVC) [15] software development pattern. The *prefuse* Action, Render and Display packages constitute the view while *prefuse* Control and Data Query packages comprise the controller. In Figure 2.6, a high level block diagram of the *prefuse* toolkit is presented and MVC design pattern elements are emphasized. *Prefuse* classifies data structures into two main hierarchies: the Tuple and TupleSet interfaces. These two interfaces incorporate

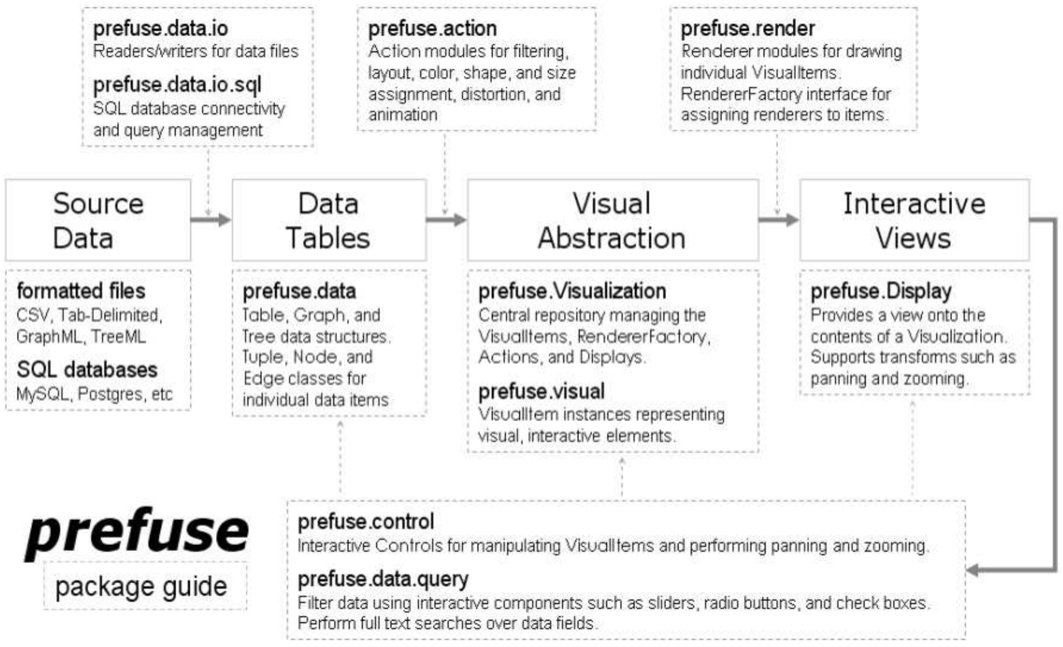

prefuse.data.io
Readers/writers for data files
prefuse.data.io.sql
SQL database connectivity and query management
prefuse.action
Action modules for filtering, layout, color, shape, and size assignment, distortion, and animation
prefuse.render
Renderer modules for drawing individual VisualItems. RendererFactory interface for assigning renderers to items.
Source Data
Data Tables
Visual Abstraction
Interactive Views
formatted files
CSV, Tab-Delimited, GraphML, TreeML
SQL databases
MySQL, Postgres, etc
prefuse.data
Table, Graph, and Tree data structures. Tuple, Node, and Edge classes for individual data items
prefuse.Visualization
Central repository managing the VisualItems, RendererFactory, Actions, and Displays.
prefuse.visual
VisualItem instances representing visual, interactive elements.
prefuse.Display
Provides a view onto the contents of a Visualization. Supports transforms such as panning and zooming.
prefuse
package guide
prefuse.control
Interactive Controls for manipulating VisualItems and performing panning and zooming.
prefuse.data.query
Filter data using interactive components such as sliders, radio buttons, and check boxes. Perform full text searches over data fields.

Figure 2.6: *Prefuse* toolkit with Model, View, and Controller highlighted [9].

collections of data structures shown in Figure 2.7 and all the concrete data structures shown in Figure 2.8 [5].

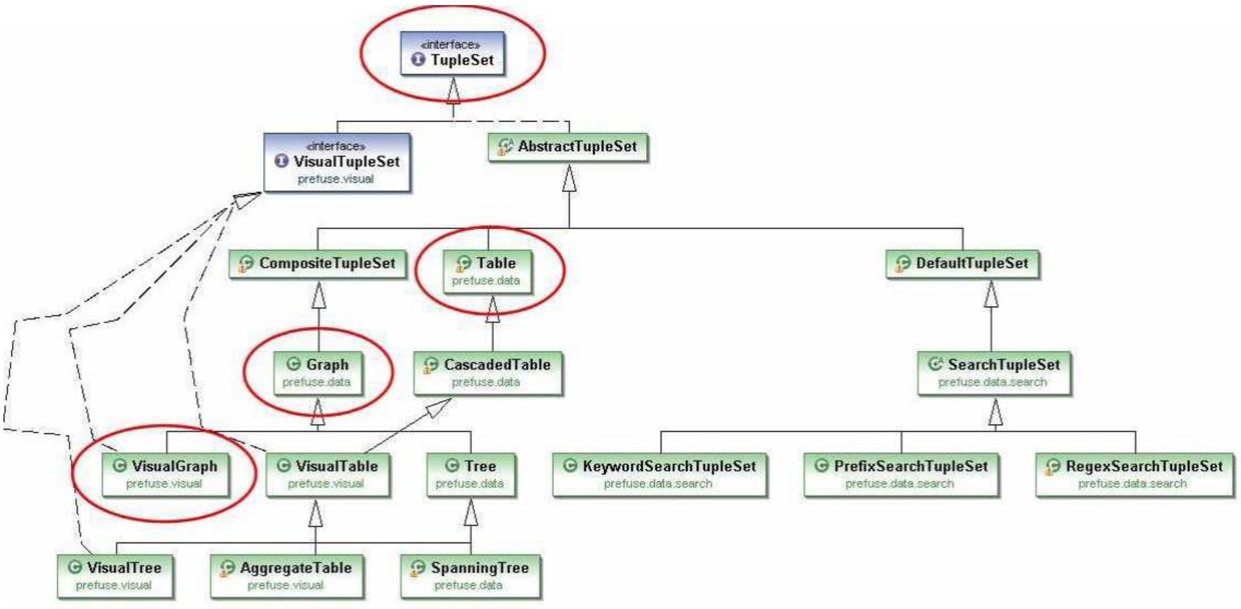


Figure 2.7: *Prefuse* TupleSet Hierarchy [5].

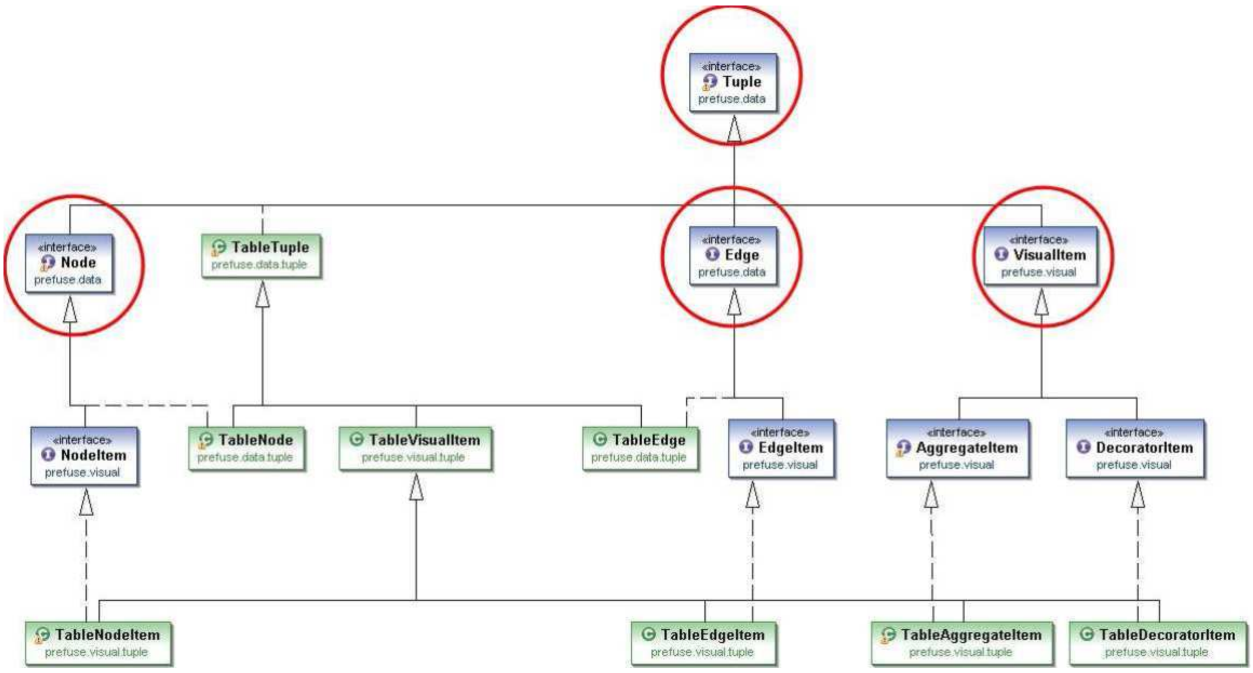

«interface» Tuple prefuse.data
«interface» Node prefuse.data
TableTuple prefuse.data.tuple
«interface» Edge prefuse.data
«interface» VisualItem prefuse.visual
«interface» NodeItem prefuse.visual
TableNode prefuse.data.tuple
TableVisualItem prefuse.visual.tuple
TableEdge prefuse.data.tuple
«interface» EdgeItem prefuse.visual
«interface» AggregateItem prefuse.visual
«interface» DecoratorItem prefuse.visual
TableNodeItem prefuse.visual.tuple
TableEdgeItem prefuse.visual.tuple
TableAggregateItem prefuse.visual.tuple
TableDecoratorItem prefuse.visual.tuple

Figure 2.8: *Prefuse* Tuple Hierarchy [5].

Prefuse will allow this research to expand the new NetViz framework by adding the node, link, and packet interaction functionality to the existing *prefuse* classes, without any additional 3rd party application. With the usage of *prefuse* visualization toolkit, a more interactive framework design can be achieved which handles the large computer and communication networks, allowing the user to interact with the current display of the framework and control the network through the visualization.

2.2.2 Mediator Tool. The *mediator* tool [12] was the subject of research conducted by Major John S. Weir, a student at the Air Force Institute of Technology. The *mediator* tool, written in the Java programming language, provides a connection between the simulation and visualization by passing data and commands, in near-real time.

For the purpose of our research, the *mediator* tool enables the new network visualization design with the potential to interact with NS-2, as it executes. In a typical NS-2 procedure, a Tcl script is written to create network scenarios. Then NS-2 is performed describing this scenario. Before running NS-2, all the events are scheduled in the Tcl script. To change an event, a user has to wait until NS-2 is done. Then the user adds desired manipulations to Tcl script and finally reruns NS-2. The *mediator* toolkit offers a near clock-time response to user interactions in the network visualization without waiting on the simulator to complete performing analysis. Figure 2.9 shows architecture between the simulation (server) and visualizations (clients).

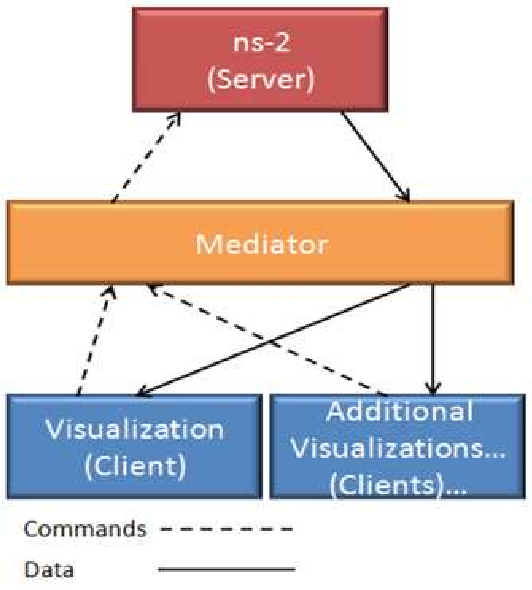
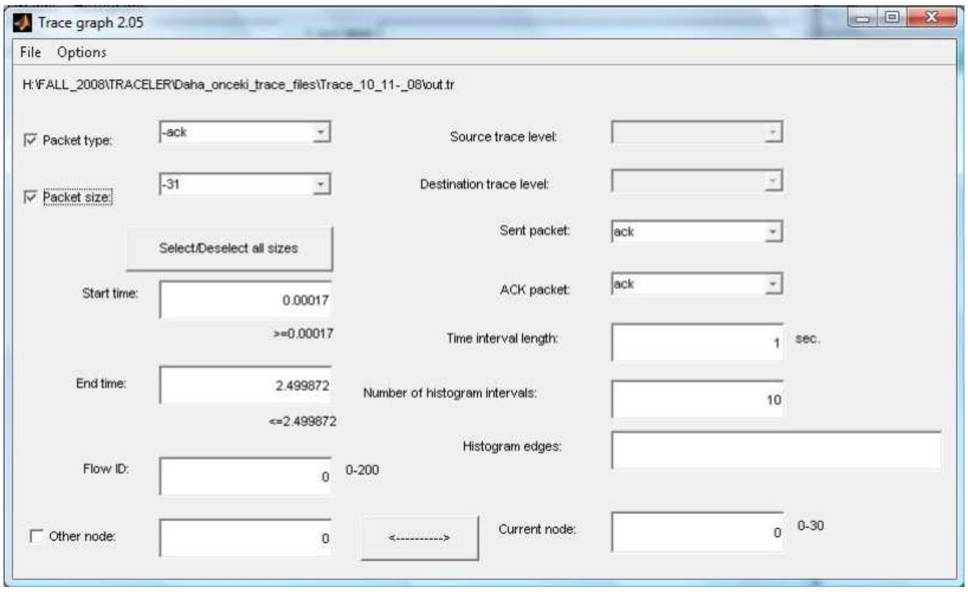


Figure 2.9: The streaming data and commands between the simulation and the visualizations. One TCP/IP connection is required per visualization, while two connections are required for the simulation [12].

Using the *mediator* tool, users will get rid of multiple iterations of a scenario to modify some features or obtain successful statistics of network events. Passing data forward and commands backward between NS-2 and the new NetViz design; the *mediator* tool will bring a kind of dynamic to the static simulation and eliminate the need for several iterations of execution.

2.2.3 Tracegraph Toolkit. *Tracegraph* [10] is a free network trace file analyzer toolkit designed for use with NS-2 trace processing. *Tracegraph* operates in Windows, Linux, UNIX and Mac OS systems. It supports any trace format for NS-2 trace files such as wired, satellite, wireless. The *tracegraph* toolkit version 2.05 has 238 2D graphs and 12 3D graphs. These graphs show delays, processing times, round trip times, throughputs and other statistics about the network. While all the results can be saved to text files, graphs can also be saved as jpeg and tiff file formats. Any file that is saved in a text file with two or three columns can be plotted using the tabs. X, Y, Z axes information of the graphs include most of the common statistical terms such as minimum, maximum, mean, standard deviation, and median. *Tracegraph* also provides script file processing to do the analysis automatically.



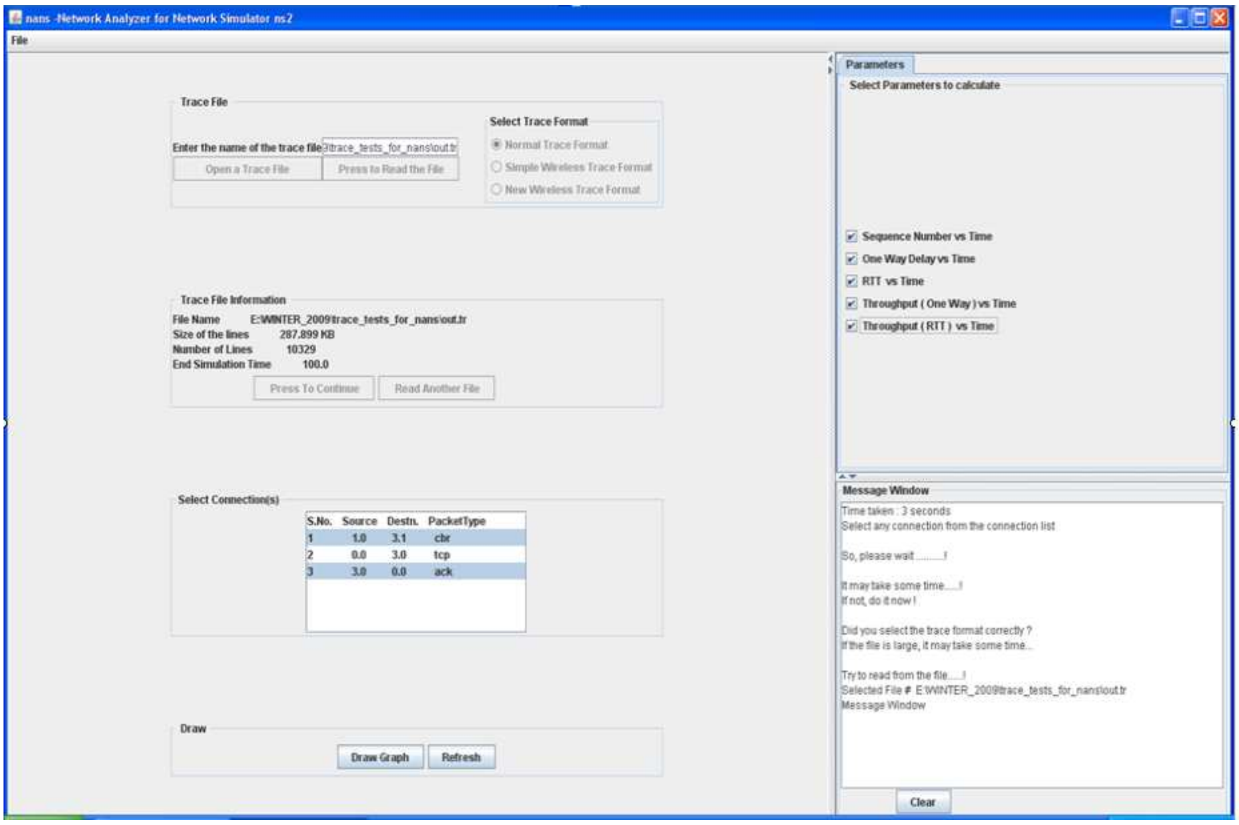
H:\FALL_2008\TRACELER\Daha_onceki_trace_files\Trace_10_11-_08\out.tr

☑ Packet type: -ack
☑ Packet size: -31
Start time: 0.00017 >=0.00017
End time: 2.499872 <=2.499872
Flow ID: 0 0-200
☐ Other node: 0
Source trace level:
Destination trace level:
Sent packet: ack
ACK packet: ack
Time interval length: 1 sec.
Number of histogram intervals: 10
Histogram edges:
Current node: 0 0-30

Figure 2.10: Graphical user interface of *tracegraph* showing a selected file to be analyzed.

Tracegraph is capable of calculating many parameters characterizing network simulation. It saves calculation results to text files and its own script files and provides many options for analysis. For the purpose of our research, 3D graphs are used to achieve a foresight among the multiple analysis options of *tracegraph,* analyzing a simple Tcl file in Chapter III.

2.2.4 Nans Toolkit. Network Analyzer for Network Simulator (nans) [11] toolkit is designed as a network analyzer for use with NS-2 trace processing. It supports any trace format for NS-2 trace files such as wired, satellite, wireless, and the new trace format. Similar to *tracegraph* tool, *nans* tool helps users of NS-2 in extracting the required data calculating parameters and thus showing 2D graphs such as sequence number, one way delay, round trip time, throughput versus time. While all the results can be saved to text files, graphs can also be saved as jpeg and tiff file formats. Trace files obtained from NS-2 will be applied to the *nans* tool to do rapid analysis of the types of 2D graphs needed for the future work.



Enter the name of the trace file: 9\trace_tests_for_nans\out.tr

Select Trace Format: [✓ Normal Trace Format] Simple Wireless Trace Format New Wireless Trace Format

File Name E:\WINTER_2009\trace_tests_for_nans\out.tr
Size of the lines 287.899 KB
Number of Lines 10329
End Simulation Time 100.0

S.No.	Source	Destn.	PacketType
1	1.0	3.1	cbr
2	0.0	3.0	tcp
3	3.0	0.0	ack

- [✓] Sequence Number vs Time
- [✓] One Way Delay vs Time
- [✓] RTT vs Time
- [✓] Throughput (One Way) vs Time
- [✓] Throughput (RTT) vs Time

```
Time taken : 3 seconds
Select any connection from the connection list

So, please wait .........!

It may take some time.....!
If not, do it now !

Did you select the trace format correctly ?
If the file is large, it may take some time...

Try to read from the file.....!
Selected File # E:\WINTER_2009\trace_tests_for_nans\out.tr
Message Window
```

Figure 2.11: User interface of *nans* showing a selected file to be analyzed.

Based on the research goals, *nans* and *tracegraph* tools provide a useful foresight in developing the statistics portion of the new NetViz framework design. The different characteristics of the *nans* and *tracegraph* graphs will guide in constructing detailed network analysis in NetViz for the future work.

2.3 Network Visualization Studies

2.3.1 P2PStudio - Monitoring, Controlling and Visualization Tool for Peer-to-Peer Networks Research. Peer-to-Peer Studio tool [16] is developed as a monitoring, controlling and visualization tool for peer-to-peer networks. It uses a centralized architecture to gather events from a peer-to-peer network and can be used to visualize network topology and to send different commands to individual peer-to-peer nodes. The tool is used with Chedar Peer-to-Peer network (see [17] and [18] for more detail) to study the behavior of different peer-to-peer resource discovery and topology management algorithms. The tool is also used for visualizing the results of the Neuro-Search resource discovery algorithm [19] generated by the Peer-to-Peer Realm network simulator which was developed to support training of neural networks for resource discovery problems.

P2PStudio is Java-based and divided into two separate programs as shown in Figure 2.12: the user interface (UI) and the server. The graphical UI connects to the server program and uses it to carry out the commands entered by the user. The graphical user interface presents the collected data visually thus making the interpretation easier compared to reading plain text log files.

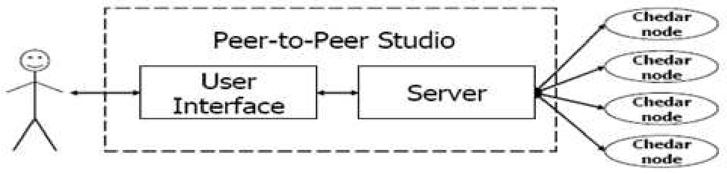
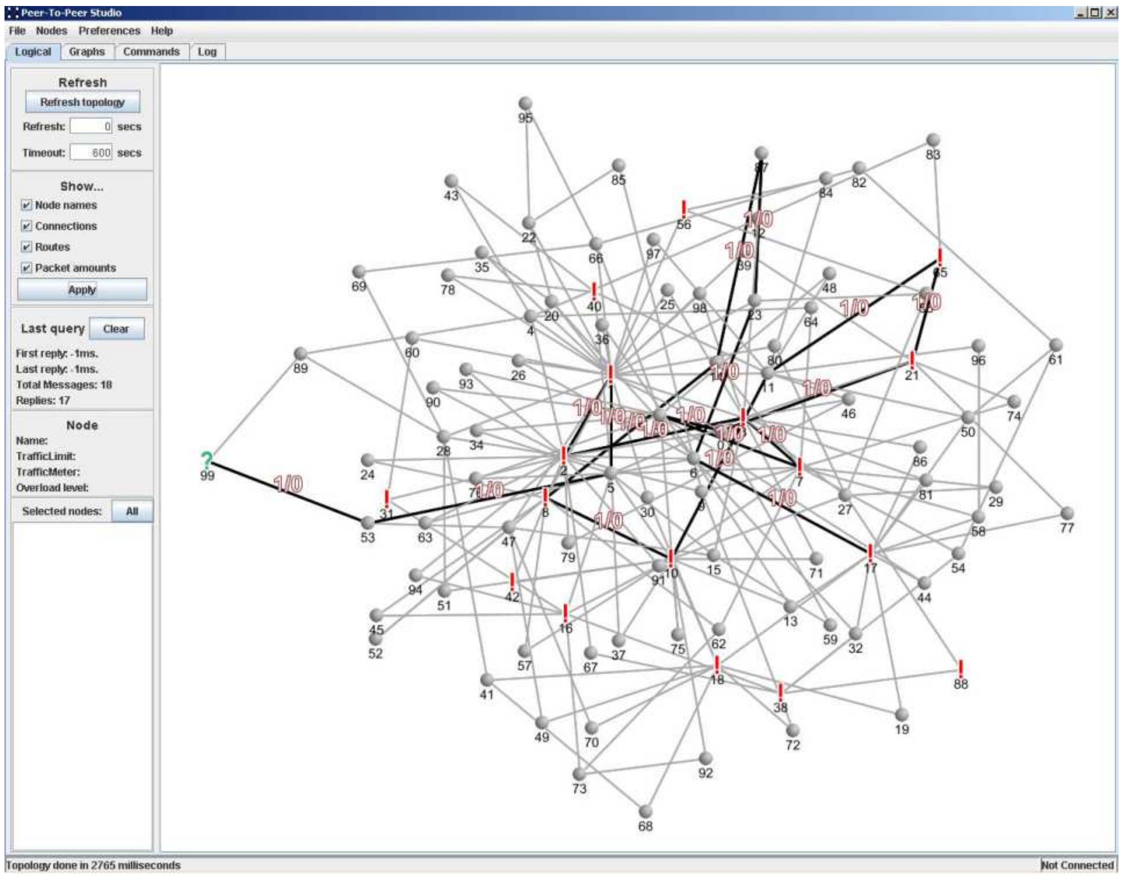


Figure 2.12: Components of Peer-to-Peer Studio showing user interface, server and Chedar nodes [16] .

The server program takes care of all of the communication between the UI and Chedar nodes. The UI communicates with the server, sends requests to Chedar nodes, displays data from the server to the user e.g., by visualizing the network topology and showing diagrams. The UI also allows the management of Chedar nodes. The server forwards the commands sent by the UI, gathers information from the Chedar network, and passes on requested data to the UI. Our research will use a similar architecture by the use of the mediator that exchanges data and commands between the simulation and visualization.

The user interface draws a logical topology of the monitored network as shown in Figure 2.13. Another feature of the UI is to show graphs of the monitoring data


Peer-To-Peer Studio

File Nodes Preferences Help

Logical | Graphs | Commands | Log

Refresh

Refresh topology

Refresh: 0 secs

Timeout: 600 secs

Show...

- [✓] Node names
- [✓] Connections
- [✓] Routes
- [✓] Packet amounts

Apply

Last query Clear

First reply: -1ms.
Last reply: -1ms.
Total Messages: 18
Replies: 17

Node

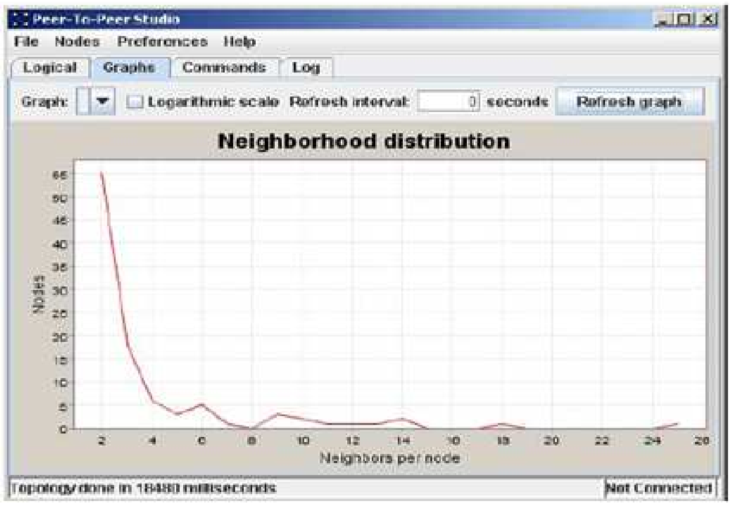
Name:
TrafficLimit:
TrafficMeter:
Overload level:

Selected nodes: All

Topology done in 2765 milliseconds

Not Connected

Figure 2.13: Topology view showing 99 nodes on the visual presentation and multiple options on the logical tab to control these nodes [16] .


Peer-To-Peer Studio

File Nodes Preferences Help

Logical | Graphs | Commands | Log

Graph: [▼] ☐ Logarithmic scale Refresh interval: 0 seconds Refresh graph

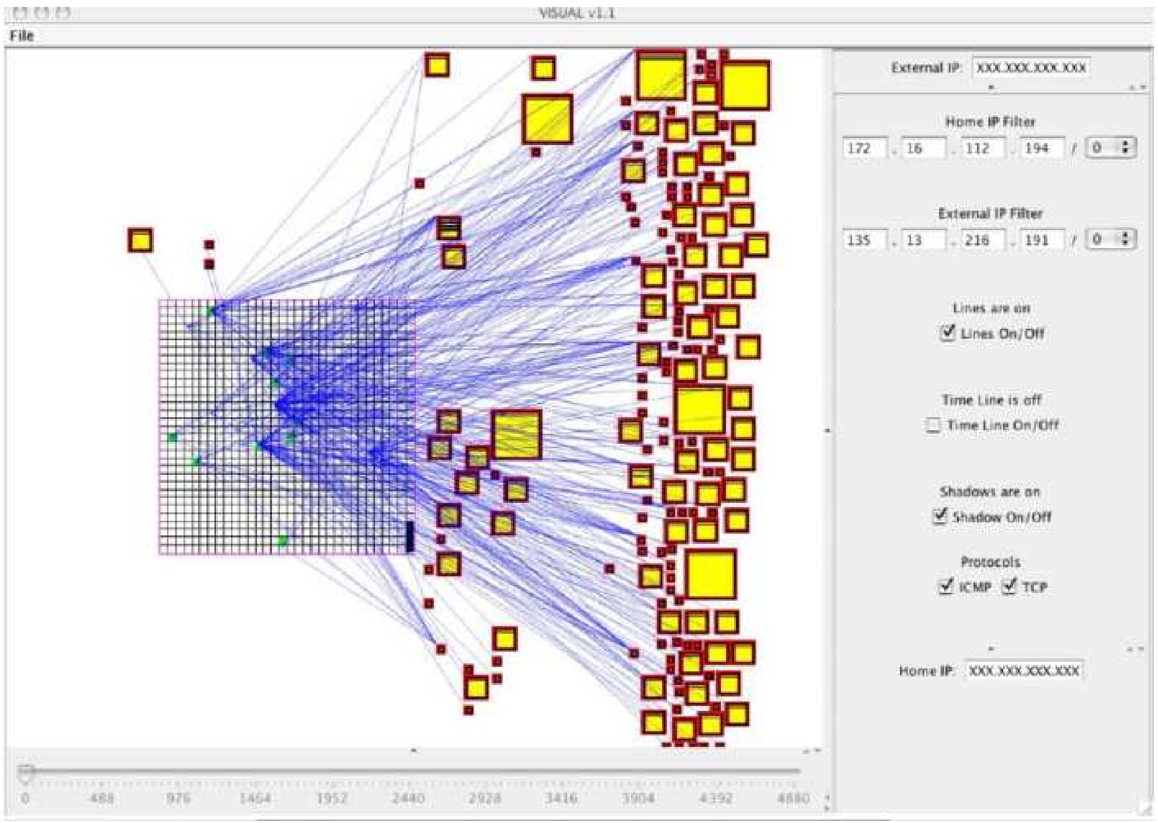
Topology done in 18480 milliseconds

Not Connected

Figure 2.14: Graph view of neighborhood distribution for a selected node [16] .

as shown in Figure 2.14. Graphs are formed by combining multiple events into a single value and are essential in network management and analysis. Plugging the various graphs such as throughput of generating/sending/receiving/forwarding/dropping packets or bits versus time and packet ID or send/receive event time versus RTT into the new NetViz design is a good contribution to our research.

2.3.2 Home Centric Visualization of Network Traffic for Security Administration. Visual Information Security Utility for Administration Live (VISUAL) [20] is a network security visualization tool. This tool allows users to see communication patterns between their home (or internal) networks and external hosts. A new computer security visualization that gives a quick overview of current and recent communication patterns in the monitored network to the users is designed and tested. While


External IP: XXX.XXX.XXX.XXX

Home IP Filter: 172 . 16 . 112 . 194 / 0

External IP Filter: 135 . 13 . 216 . 191 / 0

Lines are on — [✓ Lines On/Off]

Time Line is off — Time Line On/Off

Shadows are on — [✓ Shadow On/Off]

Protocols: [✓ ICMP] [✓ TCP]

Home IP: XXX.XXX.XXX.XXX

Figure 2.15: VISUAL view representing each home (internal) host as a small square within a larger grid that stands for the set of home hosts [20].

many tools can detect and show fan-out and fan-in, VISUAL shows network events graphically, in context. VISUAL provides insight for networks with up to 2,500 home hosts and 10,000 external hosts, shows the relative activity of hosts, displays them in a constant relative position, and reveals the ports and protocols used.

The developers of the VISUAL show a home-centric, internal vs. external perspective of the network. VISUAL displays a representation of each home (internal) host as a small square within a larger grid that stands for the set of home hosts (see Figure 2.15). A user can see which home hosts received connections from a large number of external hosts (fan-out) and which external hosts communicate with a large number of internal hosts (fan-in). This study includes a large number of events, similar to our research features. While it focuses on the network user's security which is slightly different than a platform showing the detailed network activities, it does have ideas for large scale networks.

2.3.3 Vizster Visualizing Online Social Networks. Vizster [21] is a visualization system designed and implemented for playful end-user exploration and navigation of large-scale online social networks using prefuse toolkit. The design builds upon familiar node-link network layouts to contribute customized techniques. These techniques provide increased awareness of their online community and helps exploring connectivity in large graph structures, supporting visual search and analysis, and automatically identifying and visualizing community structures. Vizster presents social networks using a familiar node-link representation, where nodes represent members of the system and links represent the articulated "friendship" links between them (Figure 2.16). In this view, network members are presented using both their self-provided name and, if available, a representative photograph or image.

The networks are presented as egocentric networks: networks consisting of an individual and their immediate friends. Users can expand the display by selecting nodes to make visible others' immediate friends as well. As shown in Figure 2.17, Vizster in X-ray mode visualizes each member's gender using color coding.

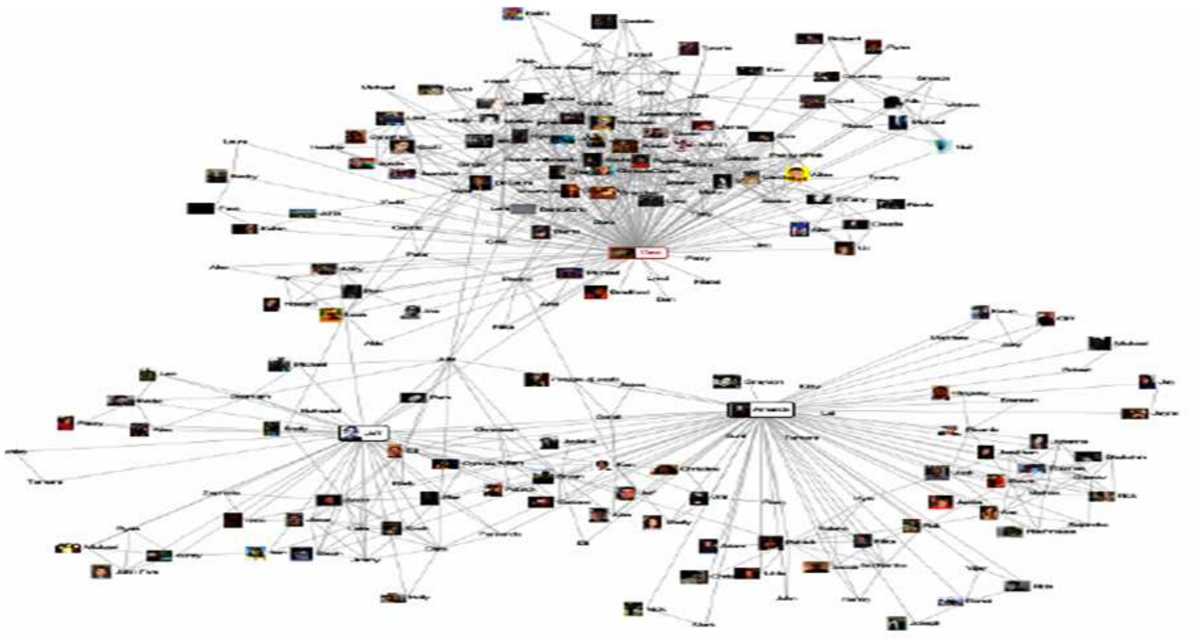


Figure 2.16: Screenshot of the Vizster visualization system depicting three intersecting social networks with the images and names of the users [21].

The Vizster design constitutes a visual environment for the exploration and analysis of online social networks, including both topological and profile data. The scale of displayed information and layout were chosen to support observed behavior and capabilities, and allow users to expand visualized networks while maintaining landmarks.

In the Vizster design, most of the interactivity features of *prefuse* such as highlighting, panning, zooming, and x-ray mode are utilized and are good examples for our approach. Since we aim to decrease the complex view of the large networks, filtering, labeling and color coding will be used which is similar to data mining and highlighting in their study. Like Vizster, in our research the scale of displayed information will be determined in accordance with the desired capabilities of the new framework design. Vizster's visual exploration of member profile is similar to our aim which is to detail the displayed node properties. Vizster's community structure is also parallel to our proposed clustering architecture.

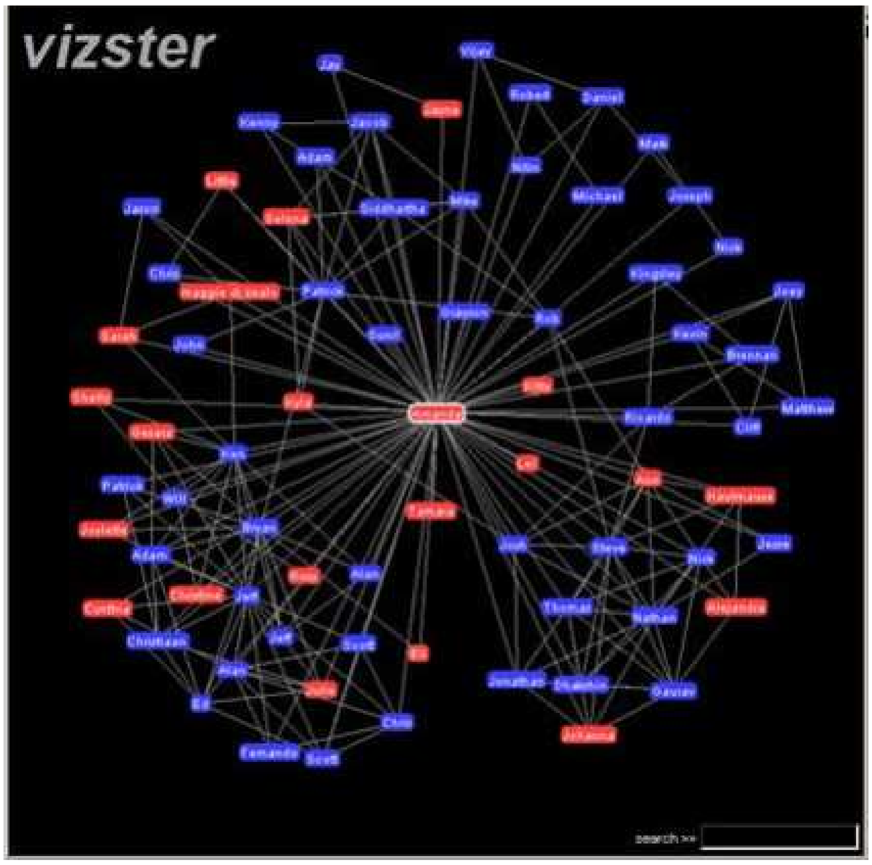


Figure 2.17: X-ray mode visualizing genders of the users. Blue shows males and red shows females [21].

2.3.4 Scalable Architecture for Monitoring and Visualizing Multicast Statistics. In this paper, Mantra [22], a tool that is developed to monitor multicast is introduced. Mantra collects, analyzes, and visualizes network-layer (routing and topology) data about the global multicast infrastructure. The two most important functions of Mantra are: (1) monitoring multicast networks on a global scale; and (2) presenting results in the form of intuitive visualizations. Mantra uses several interactive visualization mechanisms to present statistics, topology maps, and geographic properties.

Collected and processed data is used to generate useful views of various aspects of multicast. They visualize results using several tools: Otter [23], for interactive topology visualizations; GeoPlot [24], for visualization of the geographic placement of various multicast entities; and MultiChart [22], a tool that is developed to provide interactive graphing for Mantra.

The developers of Mantra use a set of static as well as interactive visualization mechanisms for presenting results. Their visualizations present both multicast activities and detailed analysis of routing problems. Mantra uses five output interfaces for presentation of results: (1) tables, (2) static graphs, (3) interactive graphs, (4) interactive topology maps and (5) interactive geo-graphical representations. Statistics are presented in the form of customizable graphs. MultiChart provides a user-friendly interface for controlling different visualization aspects of the graphs, e.g., overlaying different graphs on the same display, choosing temporal range of data, and scaling graphs.

A case study of the use of Mantra is presented to detect a routing problem, discover its cause, and evaluate its effects. The case they present pertains to a Multicast Border Gateway Protocol (MBGP) [25] routing problem that they noticed on August 21, 1999 at ORIX, in one of the routers that they collect data from. Figure 2. 18(a) graphs the number of session participants over time. There is an unusual drop in the number of sources at 1:56am on August 21, 1999. Figure 2.18(b) shows the distribution of the number of MBGP routes. There is a sharp drop, about 22.2%, which correlates with the number of participants.

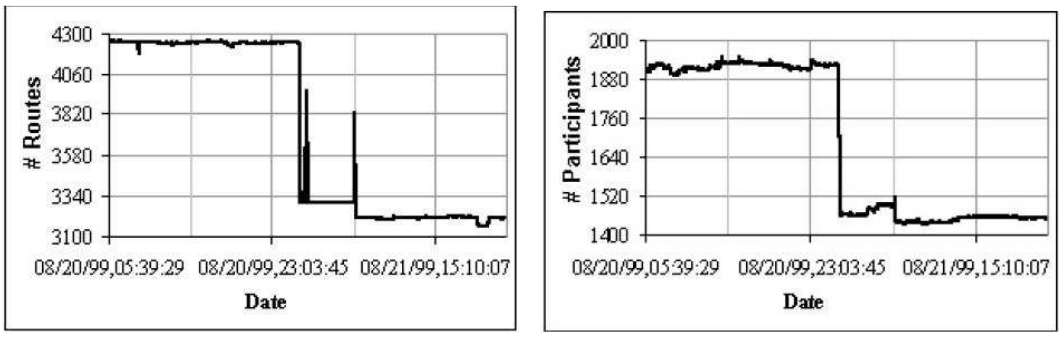
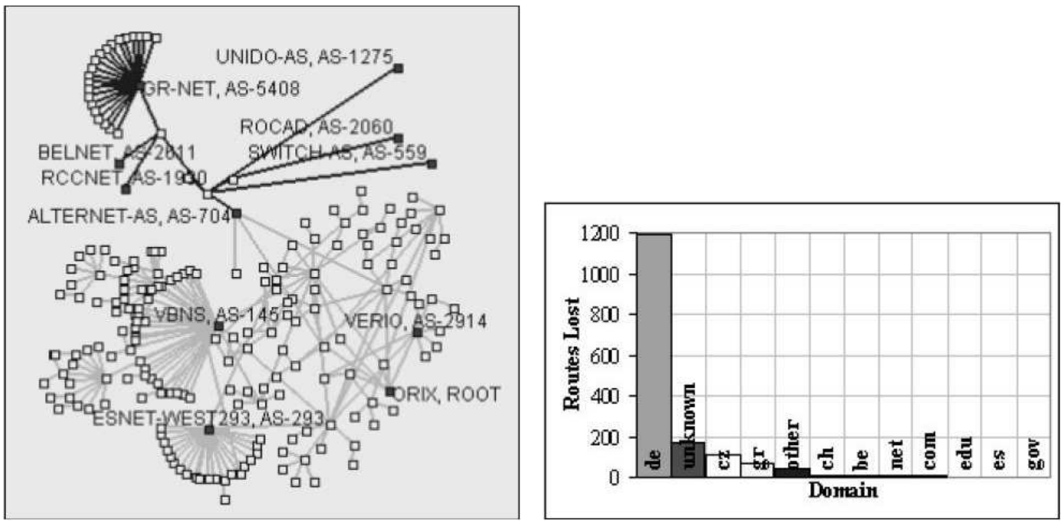


Figure 2.18: (a) Number of Participants (b) Number of MBGP Routes [22]

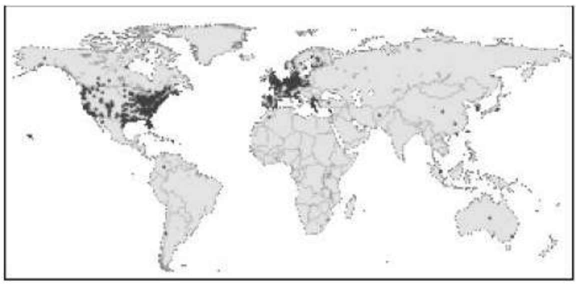


(a) (b)

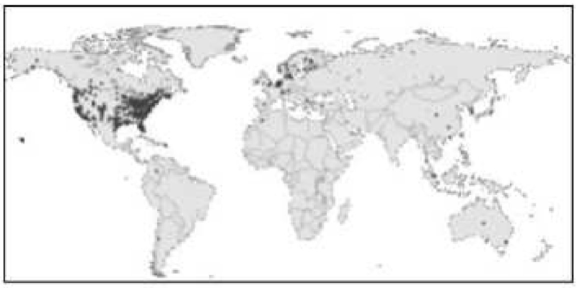
Figure 2.19: (a) Loss in MBGP Connectivity (b) Domains that Loss Connectivity [22]

Figure 2.19(a) shows a screen shot of two consecutive snapshots of the MBGP topology overlaid on the same display. Links common to both topology snapshots are in light gray; those seen only in the second snapshot are black. The bar chart in Figure 2.19(b) shows statistics about the first-order domain of the hosts. Each bar reflects the number of hosts lost from that domain. Loss in connectivity to a large number of hosts present in Europe, especially in Germany (domain name suffix "de"), Czech Republic (domain name suffix "cz") and Greece (domain name suffix "gr") is evident. Figure 2.20(a) shows geographic placement of participant hosts on a world map before the drop, Figure 2.20(b) displays scenario after the drop.

The usage of different interactive graphing and charting of various statistics guides our research by means of detailed observation of network events. Their use of interactive visualization mechanisms is similar to our research objectives. While they treat the network as a whole to detect general issues such as multicast activities and to troubleshoot routing problems, our research focuses on similar link status contingencies, packet activities, node movements and analysis of these network events.



(a) Before the loss

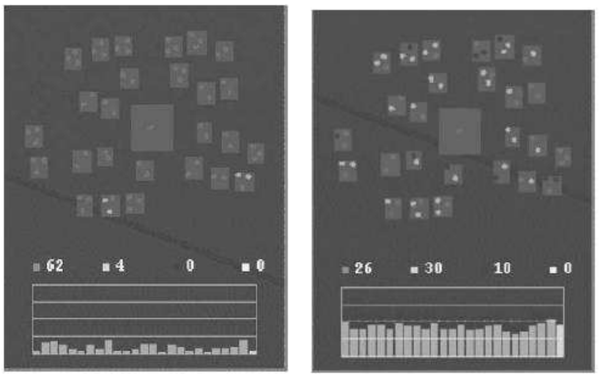


(b) After the loss

Figure 2.20: Affects of Loss in Connectivity [22]

2.3.5 Ecological interface design: a new approach for visualizing network management. Ecological interface design (EID) [26], which is a systematic approach, drawn from nuclear power plant control, for designing visualizations that uses a multi-level analysis to develop graphics designed to support problem solving and management activities. The developers of EID present an adaptation of this approach to network management and show how visualization tools can be designed. The EID tool is evaluated against an industry tool for broadband network management that is used for large networks to monitor, HPOpenView Network Node Manager(NNM) a series of detection and diagnosis tasks. As they observed slightly faster detection times with NNM, the EID tool generated faster diagnosis times and more accurate diagnoses. The authors of [26] took their tool to professional network managers for a qualitative evaluation.

They made their overview display a dedicated view in the top left-hand corner of the display (shown in Figure 2.21). This view presents a 2D image of the topological layout of all the switches and routers in the network. The left shot in Figure 2.21(a) is that of a relatively healthy network while the one to the right in Figure 2.21(b) depicts a less healthy one. A trending bar graph is shown in the bottom of the display to show network performance over time.



(a) A relatively healthy network (b) A less healthy network

Figure 2.21: Two close-up shots of the overview map of 2D image of the topological layout of all the switches and routers in the network [26]

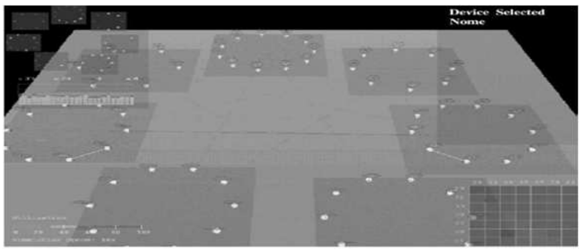
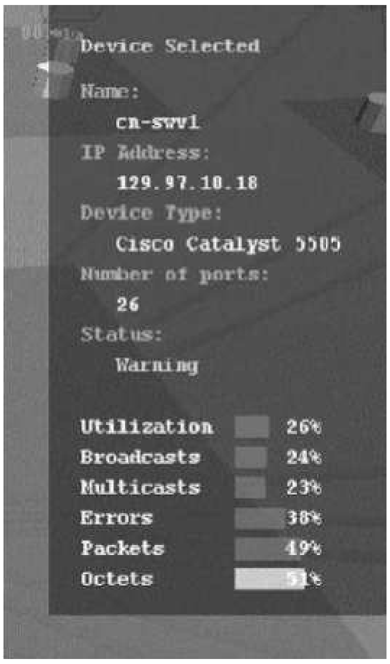


Figure 2.22: Logical view of residence network in Visual Network showing the traffic levels between and within different VLANs [26].

Figure 2.22 shows the traffic levels between and within different Virtual Local Area Networks (VLANs). The overview map not only provides displaying purposes, but also serves as a navigation aid for the main 3D view (Figure 2.22). By positioning the mouse pointer and selecting a point on the overview map, users can gain access to the corresponding area of the network.

The display that is shown in Figure 2.23(a) provides some general information on the currently selected device, such as its name, IP address, device type, and number of ports. The switch diagnosis view in NNM is shown in Figure 2.23(b) and the



Device Selected

Name:
cn-swv1

IP Address:
129.97.10.18

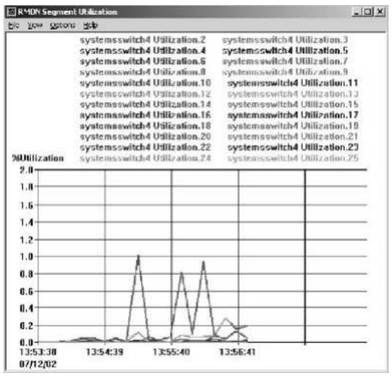
Device Type:
Cisco Catalyst 5505

Number of ports:
26

Status:
Warning

Utilization	26%
Broadcasts	24%
Multicasts	23%
Errors	38%
Packets	49%
Octets	51%

(a)



(b)

Figure 2.23: (a) The device info panel view (b) Switch diagnosis view in NNM [26]

main network view is shown in Figure 2.24. In summary, the developers of [35] observed faster detection times with NNM, yet faster diagnosis times and more accurate diagnoses with the EID display.

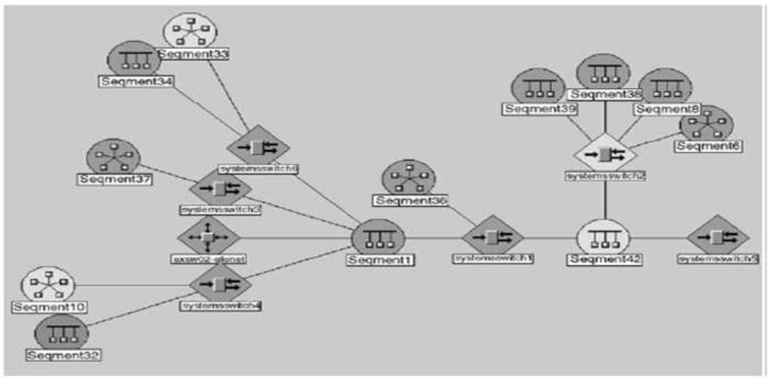


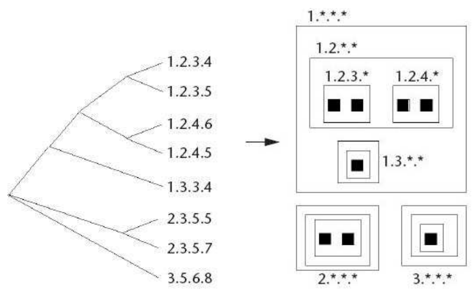
Figure 2.24: Network view from NNM [26] .

With the usage of multi-level analysis to develop graphics bar and overview map, [26] has provided a different point of view to the perspectives of our research about problem solving and management activities. While the new NetViz design should enable users to monitor the overview of network anytime, it also should allow users to pick any network object for a better observation. For the purpose of our research, using various visualization functions which provide effective access to network events, users will have enhanced ability to evaluate the network events' activities.

As the authors of [26] mentioned, the EID tool is limited in scale and the particular visualizations used may not scale effectively to a larger network situation. From this perspective, it doesn't contribute much to our research. It highlights the interaction of displays graph information.

2.3.6 Hierarchical Visualization of Network Intrusion Detection Data. The authors of [27] used a technique for visualizing intrusion-detection system log files using hierarchical data, based on IP addresses represents the number of incidents for thousands of computers in one display space. Their technique differs from other techniques in that they try to maximize the density of the information on display.

The developers of [27] group computers according to their IP addresses to form hierarchical data initially by the first byte of their IP addresses, then by the second byte, and finally by the third byte. Consequently, the technique forms four level hierarchical data, as Figure 2.25(a) shows. It visualizes the computer network's structure by representing the hierarchical data as in Figure 2.25(b), where black icons represent computers and rectangular borders represent groups of computers.



(a) (b)

Figure 2.25: (a) Hierarchy of computers according to their IP addresses (b) Illustration of visualization results of the hierarchical data [27]

The developers of [27] represent computers as small clickable icons, letting the user interface present detail on demand. Their technique applies a hierarchical data visualization technique (illustrated in Figure 2.26) that represents leaf nodes as black square icons and branch nodes as rectangular borders enclosing the icons.

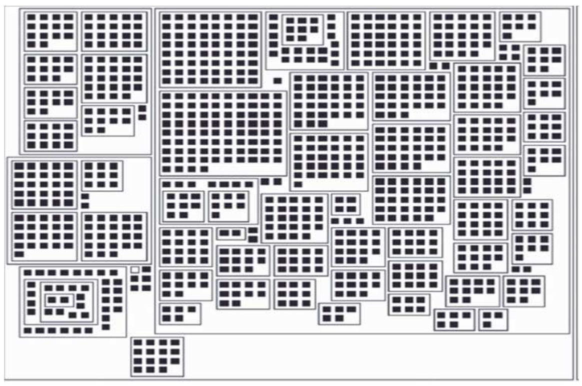


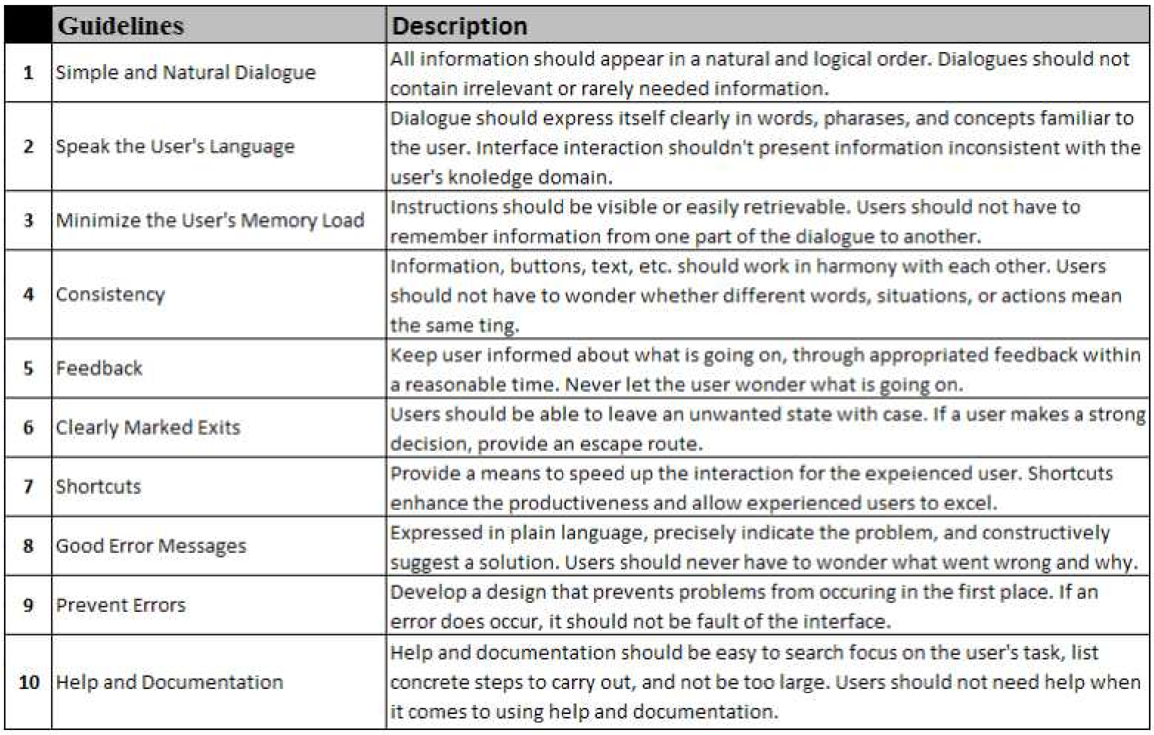
Figure 2.26: Example of hierarchical data visualization representing leaf nodes as black square icons and branch nodes as rectangular borders enclosing the icons [27].

The technique places thousands of leaf nodes into one display space while satisfying the following conditions: (1) it never overlaps the leaf and branch nodes in a single hierarchy of other nodes. (2) It attempts to minimize the display area requirement. (3) It draws all leaf nodes using equally shaped and sized icons. The unique approach of [27] contributes to our research giving the idea of structure of hierarchical order which is very significant for the military applications of NetViz.

2.4 Interactivity in Network Visualization

For a network visualization tool, interactivity is generally accepted as the most important feature by means of its usefulness. Interactivity is mainly defined as the ability to dynamically change the graphical representations and necessarily respond to user inputs [28]. “Humans are adapted for interacting with their physical environment and making continuous use of all their senses” [29]. This research explores network visualization functions and develops design specifications in Chapter III using the Heuristic Evaluation Guideline [6] . Design Specifications helped our research to build a prototype design which guides to the development and implementation process of the new NetViz design. With the usage of visualization functions and interactive techniques, the new NetViz framework design will allow users to focus on the purposed area of the network manipulating the current display.

2.4.1 Heuristic Evaluation Guideline. Heuristic evaluation [7] is a list of ten guidelines used to evaluate a user interface. It is a basic methodology for implementing the concepts of usability engineering [30] which is a field that is concerned generally with human-computer interaction and specifically with making human-computer interfaces that have high usability or user friendliness. Heuristic evaluation is one of four techniques used in usability engineering.



	Guidelines	Description
1	Simple and Natural Dialogue	All information should appear in a natural and logical order. Dialogues should not contain irrelevant or rarely needed information.
2	Speak the User's Language	Dialogue should express itself clearly in words, pharases, and concepts familiar to the user. Interface interaction shouldn't present information inconsistent with the user's knoledge domain.
3	Minimize the User's Memory Load	Instructions should be visible or easily retrievable. Users should not have to remember information from one part of the dialogue to another.
4	Consistency	Information, buttons, text, etc. should work in harmony with each other. Users should not have to wonder whether different words, situations, or actions mean the same ting.
5	Feedback	Keep user informed about what is going on, through appropriated feedback within a reasonable time. Never let the user wonder what is going on.
6	Clearly Marked Exits	Users should be able to leave an unwanted state with case. If a user makes a strong decision, provide an escape route.
7	Shortcuts	Provide a means to speed up the interaction for the expeienced user. Shortcuts enhance the productiveness and allow experienced users to excel.
8	Good Error Messages	Expressed in plain language, precisely indicate the problem, and constructively suggest a solution. Users should never have to wonder what went wrong and why.
9	Prevent Errors	Develop a design that prevents problems from occuring in the first place. If an error does occur, it should not be fault of the interface.
10	Help and Documentation	Help and documentation should be easy to search focus on the user's task, list concrete steps to carry out, and not be too large. Users should not need help when it comes to using help and documentation.

Figure 2.27: Heuristic Evaluation Guideline Articles [7].

The main reason for using heuristic evaluation instead of the entire discount usability engineering method is the lack of test users during the design process. The way to find unchanging user needs is examining Heuristic Evaluation Guideline articles and researched studies from the previous section. Knowing that user needs have the best priority in the design process, the first seven articles of Heuristic Evaluation

Guideline are evaluated by the means of meeting the user needs. The evaluation of Heuristic Evaluation Guideline mostly affected the way that we developed the design specifications. The design specifications presented in Chapter III enabled us to construct the prototype design of the new NetViz framework.

2.4.2 Network Visualization Functions. Being able to trade off among requirements for a particular situation will bring the functionality required in the new NetViz design. Some of the network visualization functions that help users perceive and interact network visualization easily are listed below. These functions need to be considered when designing the new NetViz framework.

2.4.2.1 Node and Edge Selection. For the purpose of our research the user's ability to select specific nodes and edges is important for interactivity. The ability to position nodes in a way that makes the display readable and meaningful is central to network visualization. Given that such a function exists and the position of the nodes and edges on the display are known, users can select and re-locate them. Especially, when the display is too crowded or the panning view is active and the nodes and edges are removed from the current display, this selecting and repositioning ability may be important. The new framework design will provide the user with the ability to control the network by turning the node on and off.

Once a subset of nodes has been selected, a method of representing the un-selected nodes must be chosen [31]. In the case of clustering, the selected set of nodes is the set of super-nodes or the groups themselves. According to [32] there are three possible approaches (see Figure 2.28) (a) ghosting: de-emphasizing nodes, or relegating nodes to the background (b) hiding: simply not displaying the un-selected nodes. This is also referred to as folding or eliding (c) grouping: grouping nodes under a new super-node representation. Our research will use all the approaches mentioned above in our prototype design. The ghosting approach makes nodes activated or deactivated changing their color code, etc. The usage of the hiding approach will bring “turn the nodes on and off” and “turn the labels on and off” capabilities to our

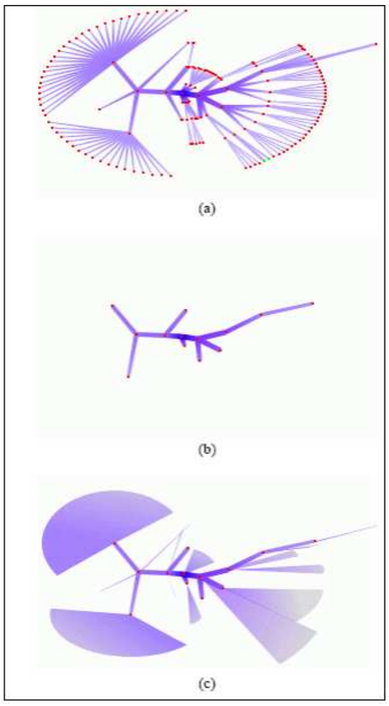


Figure 2.28: Schematic views of a tree: (a) ghosting, (b) hiding, and (c) grouping [31].

user interface. The grouping approach will serve to our design to decrease complexity of large networks, by clustering.

2.4.2.2 Clustering. Reducing the number of visible elements being viewed yields several advantages [31]. Both the clarity is improved and the performance of layout and rendering is increased by limiting the number of visual elements to be displayed. To reduce the visual complexity of a graph, researchers have applied various “abstraction” and “reduction” techniques [32]. One approach is to perform clustering. Clustering is defined as the process of discovering groupings or classes in data using previously determined semantics. Cluster analysis, grouping, clumping, classification, and unsupervised pattern recognition are among the clustering techniques referred to in the literature [31]. A mainstream technique is to create a secondary higher-level graph which includes clusters to navigate within instead of the original graph. This compound graph represents clusters with glyphs-cognitive symbols that represent common properties of cluster elements and consider them as super-nodes [31].

A common technique is to represent the clusters with glyphs and treat them as super-nodes in a higher-level or compound graph, which researchers can now navigate instead of the original graph [33]. One novel solution is to exclude the edges and position the nodes indicating their connectivity and thus eliminate the problem of edge-crossings and reduce visual clutter [34]. This solution eliminates the problem of edge-crossings and reduces visual clutter. If the same clustering logic is repeatedly applied to re-cluster the already clustered super-nodes, then this process is defined as hierarchical clustering [31]. It's shown in Figure 2.29 where each cluster is represented as a node in the tree.

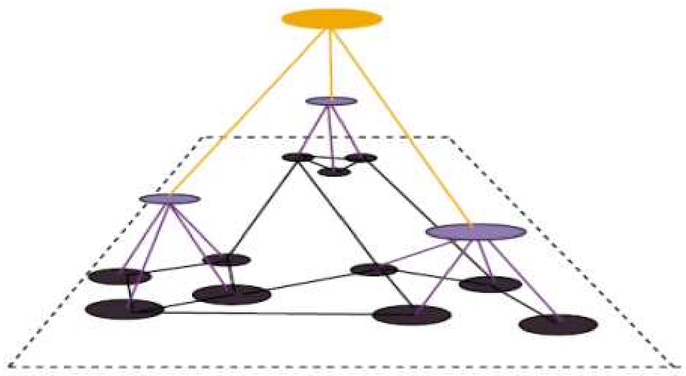


Figure 2.29: A structure induced by hierarchical clustering [31].

2.4.2.3 Metric Value. It is important to use the numerical values associated with the nodes for providing user interactivity. A node metric criterion can be used to determine an abstract property about any of the nodes for comparison of the nodes with each other and to achieve a ranking in between [31]. A numeric computable function may count for such a metric criterion. The bandwidth of the links between the nodes is a computable numeric value and can be computed using the trace file. In the current visualization framework, this is shown as follows: Nodes that have higher bandwidth links in between appear closer on the graph and nodes that have lower bandwidth links are pulled farther apart. But, in networks including high numbers of nodes, it is beyond the ability of the human eye to detect such a detail. Metrics can also be used to implement search or filtering, where a certain threshold is determined and the elements with metric values above this threshold is emphasized. Clustering can be done by forming groups of elements according to their metric value.

2.4.2.4 Filtering and Search. Filtering and search are among the functions that can be carried out utilizing clustering [31]. Both filtering and search provide optional detailed display and necessary for the user interaction. In visualization literature, filtering and search usually mean the opposite of each other. While filtering usually means excluding an element or group of elements, highlighting of such elements or groups of elements is meant by search. In both filtering and search the first step is partitioning the elements into two or more groups, and the second step is emphasizing or de- emphasizing, one of the groups, according to the case. This research utilizes filtering for the purpose of amplifying the user's comprehension by simplifying the visual complexity of the large networks.

2.4.2.5 Annotation. Annotation on the nodes and edges yields more meaningful displays and thus amplifies comprehension [13]. The purpose of the visualization determines the proper amount and type of annotation. If the main purpose is to simply present a logical relationship among the nodes (e.g. during structural data presentations) then relatively little and simple annotation is required. In some cases, we only need current relative information, such as "big" and "small" or functional type. In such cases size, color and/or shape may be effective. Finally, sometimes quantitative information, such as length, may be required. While annotation is useful, it requires space on the display and thus we trade it off against readability and scalability.

2.4.2.6 Dynamic Display. Much of the above-mentioned functionality implies user interactivity and control over the display. This ability to dynamically change the display provides the user with many advantages and decreases the requirements of the display software. Moreover, it authorizes the user to examine what is being displayed, which is principal to visualization. Taking into account the currently available memory and computing power and the present software libraries which are able to implement user controls, it is valuable to make a dynamic display and provide a complete GUI. Anyhow, we may still want to get as much automation as possible. Consider a user laying out a relatively large network manually. For the best case, it will be boring and monotonous. For the worst case, it will be beyond most users' capacity and error prone. Besides dynamically responding to user inputs and thus enabling user interaction, visualization software may be incorporated with other software, such as analysis algorithms. In such cases the output of the analysis might also be displayed graphically to boost comprehension [13].

2.4.2.7 Zoom and pan. Zoom and pan are conventional visualization tools, which are essential especially while large graphical structures are being explored. Using zoom and pan aliasing problems could be prevented. The operation of zooming is not difficult to realize. In fact, the technical difficulty is rather related to designating a proper level of detail and a sort of clustering, to sub-graphs [31]. Functional purpose displays are often sort of overview displays that demonstrate system performance versus various objectives [13]. An overview of the system state would be handy to be viewed and let the network manager rapidly identify network failures in any region of the network. To serve this purpose, the "view category" would include an overview option on the new user interface of the network visualization.

2.4.2.8 Color Coding. Color is very important aspect in developing a user interface. "With respect to learning and comprehension, color is superior to black-and-white in terms of the viewer processing time and emotional reactions, and there is a difference in a viewer's ability to interpret information" [35]. Emphasizing required information in a natural way, color increases comprehensibility and thus reduces errors of interpretation.

2.5 Summary

Discussing related network interface studies, exploring interactive techniques and examining network visualization functions, this research presents a prototype design which is described in the next section. To determine interactive techniques, we evaluated Heuristic Evaluation Guideline which is a basis for design specifications presented in Chapter III. Many of the techniques and functions presented in this chapter are used to create the prototype design. The prototype design guides to our research by means of development and implementation of the new NetViz design.

III. The New NetViz Framework Design and Implementation

This chapter outlines the research methodology used for the development of the new NetViz framework design, which describes prototype design, an overview of NS-2 result processing, implementation process, visualization/simulator interaction and demonstration of two simulation execution scenarios. Utilizing different tools, interactive techniques and visualization functions represented in Chapter II, this research applies some of the ideas that are obtained from researched studies to the prototype design. Clustering, labeling, color coding and selecting nodes capabilities facilitated in the prototype design are designed to assist in accessing network objects and events. These functions support four tabs consisting of buttons, menus, and sliders which provide control over the network objects for different tasks.

The prototype design is used as a guide for the implementation process of the new NetViz framework design. The overview of NS-2 result processing introduces flow of events for a NS-2 Tcl file, including a simple OTcl script which is used to obtain 2D and 3D graphs from post processing tools, *nans* and *tracegraph*. The implementation section describes object interaction functionality that is added to the actual classes to expand NetViz framework for the new capabilities. It's projected to achieve an interaction between the new NetViz and NS-2 utilizing *mediator* tool. Lastly, simulation execution scenarios are demonstrated.

3.1 Prototype Design

Humans can not easily process large volumes of packet traffic at a glance. A shortage of existing NetViz user interface was the lack of capability of handling large and complex network scenarios. Using various interactive techniques this research produced a prototype design of NetViz user interface to make the complex data set understandable. The primary goal of the prototype design is to explore conventional intera ction and display methods for optimizing and controlling large network scenarios.

In the process of creating the prototype design, the Heuristic Evaluation Guidelines [6] helped us to develop design specifications. User needs and interactive functions to meet these needs are described in the specifications. The major purpose of the prototype design consisting of these interactive functions is to ease the development and implementation process of the new NetViz framework design. The prototype design helps to determine the ways to facilitate the complexity of large volume of network traffic and allows rapid perceiving of the interface components which serve to an effective NetViz user interaction.

3.1.1 Design Specifications. A set of design specifications were developed evaluating the first seven articles of Heuristic Evaluation Guidelines discussed in Chapter II. These specifications include identification of current and future user needs and the ways to meet these needs with researched interactive functions. Before constructing the prototype design, the specifications provided a basis for the standards of interactivity and guided the development process defining visual style of the user interface. Providing consistency among the several interface components, these specifications insured the design complied with the principles of human computer interface [36]. The design specifications are listed and discussed below:

1. Users should be kept informed about the latest status of the network and the network events as the visualization executes. This can be provided by using additional pop-ups, windows, tabs, info fields or refresh buttons.
2. Users need to have a control of network objects and events as they occur in a specific time. A major factor for controlling large networks is to use the time control to speed up or slow down the time to give the user the ability to manage the speed of visualization. A few buttons can be included for this purpose.
3. The user interface's windows and visual view should give the user enough information about the network components without overwhelming them with meaningless phrases or unnecessary crowding of objects. Visual perception of users varies low to high depending on the display or visual representation quality.

To make information appear in an understandable way, clustering, labeling and color coding interactive functions can be used.

4. The design should minimize the user's memory load by making objects, actions, and options visible in different ways. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the network should be visible or easily retrievable whenever appropriate. Sliders to change the view size of items, filtering, tradeoffs can be used for this purpose. The user should be the best decision maker for these functions, having as much control as possible over the display.

5. Serving multiple purposes of visualization applications, assistance tools may help the development of the user interface. Even though these tools are not used in the implementation process, they're important in terms of providing foresight into the development of the new NetViz framework design. For the analysis of a simple network scenario *nans* and *tracegraph* tools are used (See Chapter II).

6. A balance should be built between the user control abilities and workload without overwhelming the user. To give the user the ability to control execution, the ways to make long processes shorter should be examined. Distributed execution of the system using server-clients logic on several hosts can be an option. Thus, the user on the client visualization side can only deal with the user interface without taking care of controlling the server side.

These specifications are taken as a guide in the prototype design process. They enabled us to make a number of decisions to form the best prototype. These decisions and features of the prototype design are discussed in the following sections.

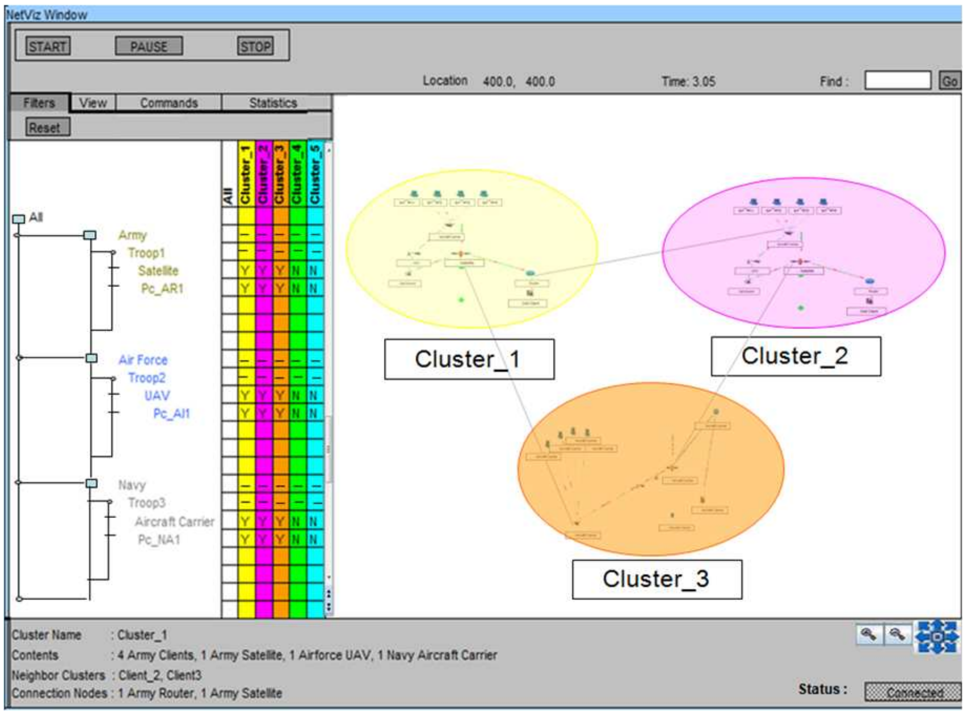
3.1.2 Four Task Areas. Considering large network scenarios, there is a need for settle on different areas to assign a number of network tasks for the prototype design. Determination of these tasks related to the observation of the network leads to an understanding of the user needs which were involved in the design specifications in the previous section. Summarizing briefly these needs, users need to have capability of controlling, optimizing, changing and analyzing the network. Because dynamic interactions support both the perception and management ability of the user through the network, many interactive functions were used to meet these needs in the new design prototype.

For the purpose of our research, the user needs are consulted to determine four task areas which are filters, view, command and, statistics. Each task amplifies the perceptual process enabling the user to have faster cognitive interaction. The four tasks formed the basis for the prototype design in terms of observation of networks.

3.1.3 Tabs. In the beginning of the design process, there were a few conflicts about the four task's exhibition style and position. Widgets for displaying tasks in separated windows such as pop-ups are compared versus tabs. Knowing that the network event animation visual representation should be displayed on one part of the screen to keep users informed, separated windows cause a struggle to maintain a flawless integrity of the visual presentation. Therefore the separated windows which cover the screen in an unwanted way are eliminated for the prototype design.

Tabs are selected as an interactive technique, reviewing related studies and discussing similar visual representations. Allowing stacked windows, tabs offer users with a panel and network event animation on the same screen and at the same time. Thus, users are to not only see the network event animation screen on the right hand side, but also do tasks for a specific object moving between tabs. All tabs are on the left hand side of the NetViz window and the active tab always is displayed in the same position (left hand side of the window again). These tabs will be discussed in the next sections.

3.1.3.1 Filters Tab. When the network is so complex that the understanding the structure of this network by eyes alone is hopeless, it is a challenge to find and analyze a specific network item. This challenge is eased by using filters which are shown in Figure 3.1.



Location 400.0, 400.0
Time: 3.05

	All	Cluster_1	Cluster_2	Cluster_3	Cluster_4	Cluster_5
All						
Army		I	I	I	I	I
Troop1		I	I	I	I	I
Satellite		Y	Y	Y	N	N
Pc_AR1		Y	Y	Y	N	N
Air Force		I	I	I	I	I
Troop2		I	I	I	I	I
UAV		Y	Y	Y	N	N
Pc_AI1		Y	Y	Y	N	N
Navy		I	I	I	I	I
Troop3		I	I	I	I	I
Aircraft Carrier		Y	Y	Y	N	N
Pc_NA1		Y	Y	Y	N	N

Cluster Name : Cluster_1
Contents : 4 Army Clients, 1 Army Satellite, 1 Airforce UAV, 1 Navy Aircraft Carrier
Neighbor Clusters : Client_2, Client3
Connection Nodes : 1 Army Router, 1 Army Satellite
Status : Connected

Figure 3.1: Screenshot of the prototype design showing filters tab on the left hand side with in a table format using a tree organization consisting clusters in various colors and visual presentation of 3 clusters with labels on the right hand side.

The filters tab includes a hierarchical order displays a tree representation of the hierarchical organization of nodes on the left hand side, filter settings in a table format on the right hand side, and "Reset", "Labels On/Off", "X-Ray View" buttons. The components of the filters tab are described in the following list:

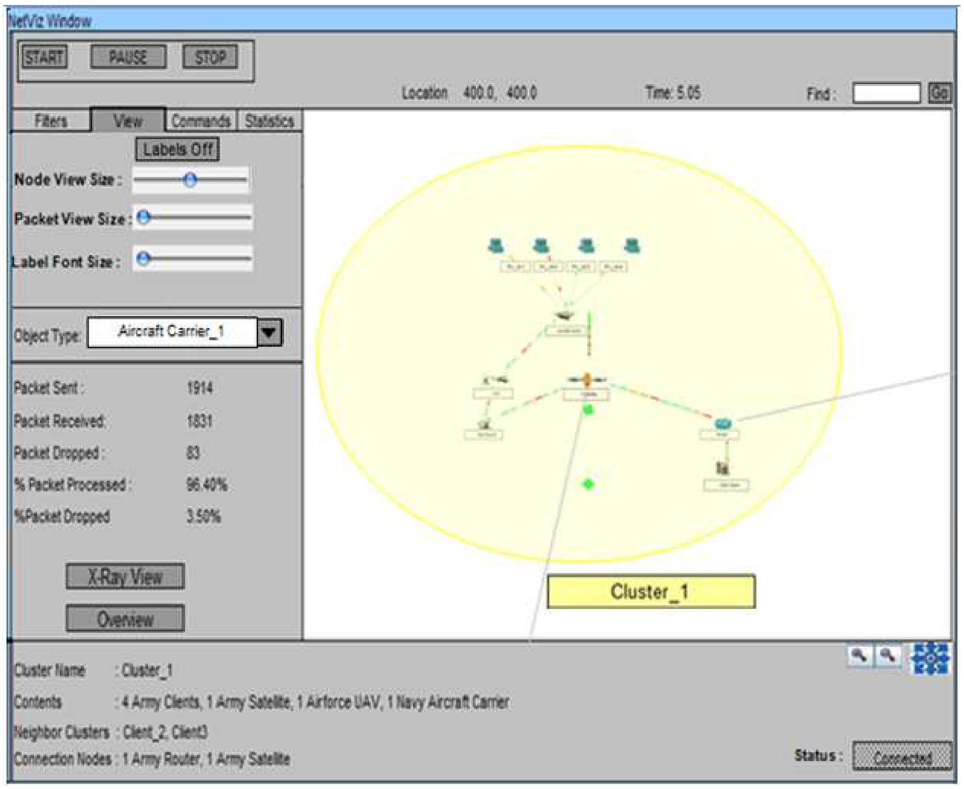
1. *Hierarchical Order.* The hierarchical order shows all the military components of the clusters on the left hand side of the filters tab. Clustering interactive function is used as a reduction technique as mentioned in the design specifications. For the case depicted in Figure 3.1, there are different military service departments such as Air Force headquarters, Navy headquarters, Army headquarters and each of them has a troop, which are named “Troop1”, “Troop2”, or “Troop3”. Sub-rows which indicate the other objects are located under these troops. With

the usage of annotation network visualization function which is identified as labeling in the design specifications, these names are displayed next to the image of each node on the design prototype's visual presentation. To display the labels on the screen, a set of modifications are done in the NS-2 files as described in section 3.4. The aim for doing this is to enhance the user perception overlaying real world scenario information.

2. *Filter Settings.* are presented in a table format using a tree organization consisting of clusters in various colors. The semi-transparency in colors allows users to follow the network objects and events easily. Like clusters, each military department is given its own color such as green for Army, blue for Air Force, gray for Navy. As mentioned in Chapter II, color coding function is utilized putting these colors in the background of the label text.

   There are rows which belong to different clusters in the table. The table can have additional rows depending on how large the network is. The filters tab has two states, that can be set, "Y" means display the item; "N" means do not display the item. As shown in Figure 3.1, there are five clusters presented in the columns. Since the 4th and 5th clusters indicated as "N" for each military component of these two clusters, there are 3 clusters visible which are yellow, purple and orange on the display. Each cluster includes nodes such as UAV, satellite and aircraft carrier. Utilizing the table, users both can depict all the objects included in clusters or display specific objects selecting them on the table.

   "Army", "Air Force", "Navy" are at top of the hierarchy as main rows. The main row can include additional filter rows which may be either representative main row or sub-row. These main rows have differing values by displaying the state as a hyphen ("-") on the table. When setting a filter entry on a main row, all rows contained by the main row will be set when the filter entry is changed. The main rows of the table can be expanded to reveal additional filters that may be set. The filters tab also includes a "Reset" button. Clicking on this button user can reset all the settings done before and see the “Y” state as a default value in all the rows.



Location 400.0, 400.0

Time: 5.05

Find :

Object Type: Aircraft Carrier_1

Packet Sent :	1914
Packet Received:	1831
Packet Dropped :	83
% Packet Processed :	96.40%
%Packet Dropped	3.50%

Cluster Name : Cluster_1
Contents : 4 Army Clients, 1 Army Satellite, 1 Airforce UAV, 1 Navy Aircraft Carrier
Neighbor Clusters : Client_2, Client3
Connection Nodes : 1 Army Router, 1 Army Satellite

Status : Connected

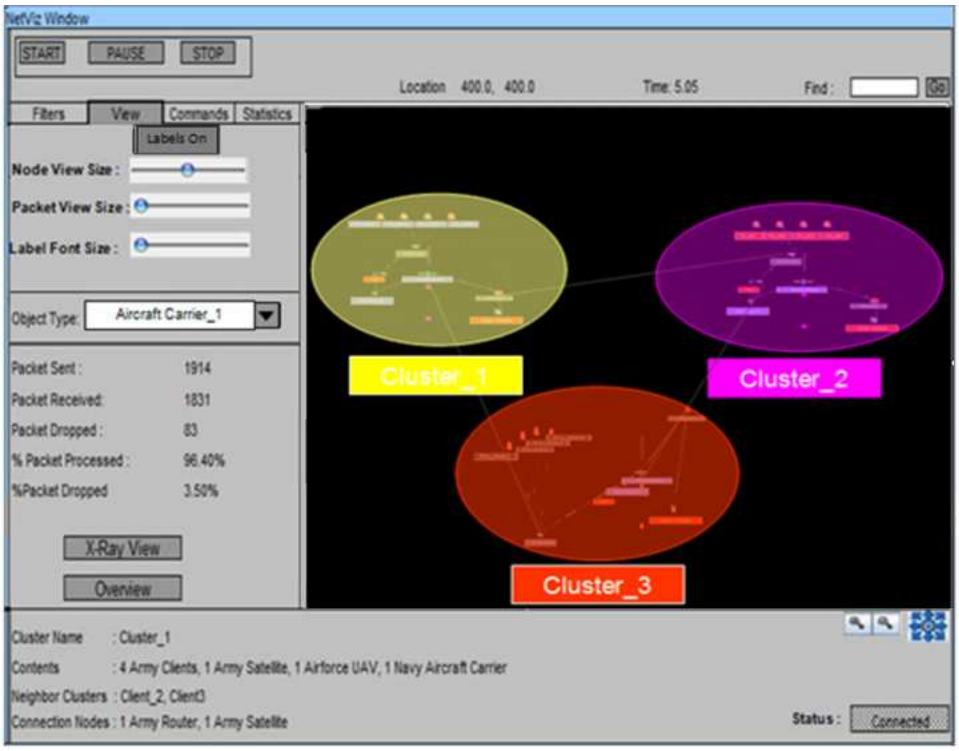
Figure 3.2: View Tab after the node view size slider has been dragged to the right.

3.1.3.2 View Tab. The view tab provides a variety of options for changing the look and feel of the network objects, labels and packets. With the usage of pull-down menu included in this tab, the user can pick any network item from the list and observe the information of this object. The following is a listing of each of the areas available in the view tab, with a brief description of what service the area provides:

1. *View Size.* Adjustment of the of network item’s view size enables the user to highlight needed items and reduce the visual complexity. Knowing that depiction of nodes using images makes users aware of node types and attributes, a slider bar is facilitated which allows users to make these images bigger or smaller. The reason for using sliders is to explore various states of the network

items for pre-determined values in a semi-automatic way without overwhelming users as mentioned in the design specification. Restricting the range of possible values, the slider gives an indication of variable's value. To change the current size of the items, the user can click on the bar with the left mouse button and drag the cursor along the length of the slider. Same type of slider bars are also facilitated for packet view size and label font size.

In the case depicted in Figure 3.2, node view size slider is dragged to the right, while there is no adjustment in the other sliders. By doing this all node view sizes are increased. As shown in Figure 3.2, increasing or decreasing node view sizes proportionally reduces complexity of the display. Moving the slider to the far left reverts the display to the initial state of the view of items. Users can use 3 sliders in different combinations until the visual presentation of network "look right" to them.



Location 400.0, 400.0

Time: 5.05

Object Type: Aircraft Carrier_1

Packet Sent : 1914
Packet Received: 1831
Packet Dropped : 83
% Packet Processed : 96.40%
%Packet Dropped 3.50%

Cluster Name : Cluster_1
Contents : 4 Army Clients, 1 Army Satellite, 1 Airforce UAV, 1 Navy Aircraft Carrier
Neighbor Clusters : Client_2, Client3
Connection Nodes : 1 Army Router, 1 Army Satellite

Status : Connected

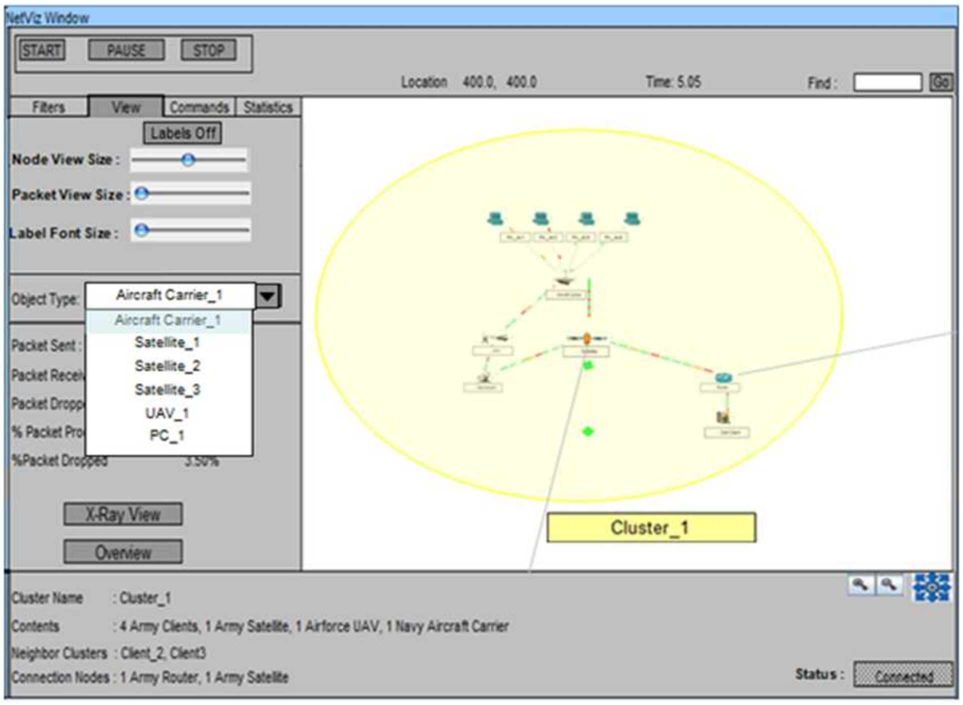
Figure 3.3: A snapshot of prototype design window view filters tab in X-Ray View mode.

2. *Pull-Down Menu.* For the large networks scenarios, the “find” field may not give enough ability to explore the network in detail. To type the name of the object in the “find” field, users need to find this object on the display. The interactive techniques are researched to provide ease in use. We came up with the usage of a pull-down menu as an additional exploration tool which includes all the network objects. Similar to filters tab, all the items appear in a pull-down list which allows the users access the desired object. The usage of pull-down menu strengthens our design giving the ability to access the desired object. In the case depicted in Figure 3.3, once the aircraft carrier 1 is selected from pull-down menu, the information about it appears below this menu.

3. *Information Regarding Network Objects.* Figure 3.2 and Figure 3.3 illustrate the information about the selected item by showing packet receives, drops, and receive/drop percentages below the pull-down menu. Depending upon which object the user selects, the field that includes information about this object will appear in the same area. In addition to selection the item from pull-down menu, user also can select the item from the picture representation using the filters tab or zooming, panning facilities.

4. *Buttons in the View Tab.* The view tab also includes four buttons, which are “Overview”, “Toggle”, “X-Ray Mode View”, and “Labels On/Off”. “Toggle” button allows users to change the link status between up and down. If the zooming and panning are performed, users can display the default view of network objects clicking on “Overview” button.

   When users click on ”X-Ray” mode button, the background of the display turns black to clearly highlight the label colors which indicate military departments or clusters. Figure 3.3 shows prototype design in X-Ray mode visualizing the network events and network objects in 3 cluster’s structure. It’s easily recognizable the colors that are coded for the forces and clusters.

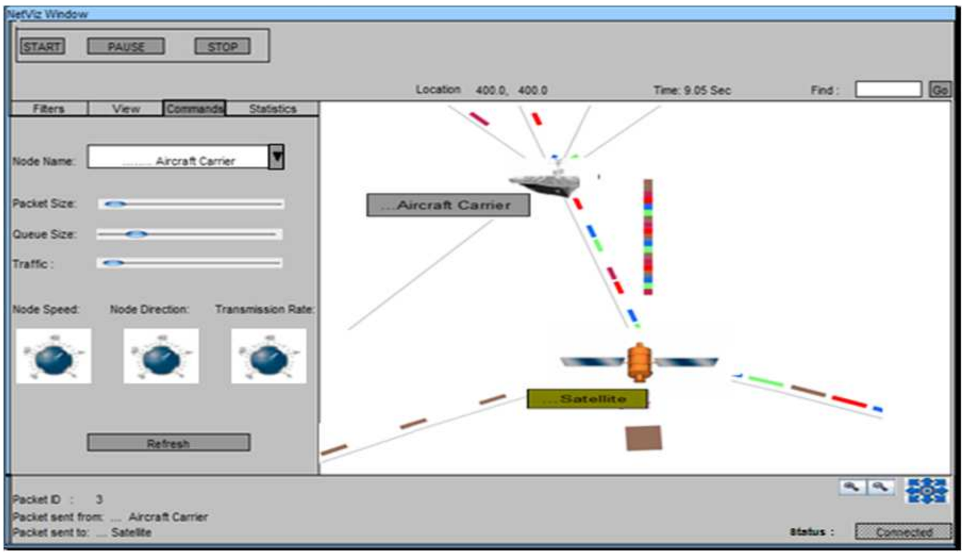


Location 400.0, 400.0
Time: 5.05
Object Type: Aircraft Carrier_1
%Packet Dropped 3.50%
Cluster Name : Cluster_1
Contents : 4 Army Clients, 1 Army Satellite, 1 Airforce UAV, 1 Navy Aircraft Carrier
Neighbor Clusters : Client_2, Client3
Connection Nodes : 1 Army Router, 1 Army Satellite
Status : Connected

Figure 3.4: View Tab showing the pull-down menu to select network items from the list.

The "Labels On/Off" button has tradeoff functionality which refers to hiding method discussed in Chapter 2. Users can make the labels invisible or visible clicking on this button (See Figure 3.4). This research utilizes tradeoff buttons for the purpose of amplifying the user's comprehension by simplifying the visual complexity of the large networks.

3.1.3.3 Command Tab. The command tab provides a pull down menu and multiple options for manipulating the network. The pull down menu gives the ability users to select an object from the list to change the network simulation. Then, using the sliders below this menu, users can send commands such as change packet size, queue size, traffic to the simulator. Since the dials have the same logic in the representation of interactive techniques, they are utilized to perform node speed, node direction and transmission rate commands (See Figure 3.5). For the purpose of our research, users will see the effects of changes made to the simulation in near clock time using these commands.



Location 400.0, 400.0

Time: 9.05 Sec

Node Name: .... Aircraft Carrier

Packet Size:

Queue Size:

Traffic :

Node Speed:

Node Direction:

Transmission Rate:

Packet ID : 3

Packet sent from: ... Aircraft Carrier

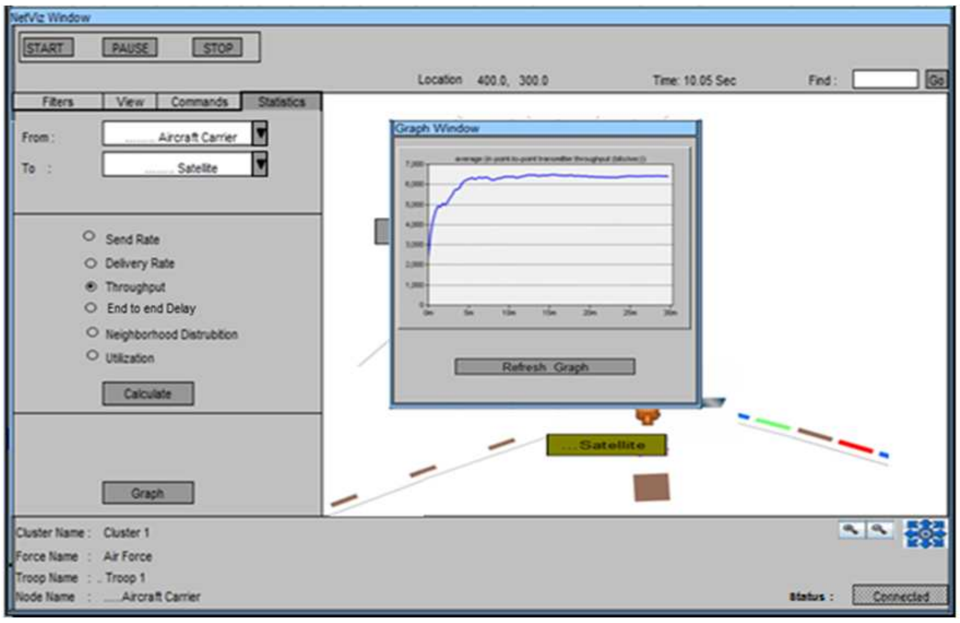
Packet sent to: ... Satellite

Status : Connected

Figure 3.5: A snapshot of prototype design showing the command tab including a pull-down menu and commands represented as sliders and dials.

3.1.3.4 Statistics Tab. The statistics tab includes two pull-down menus, the statistics such as send rate, delivery rate, throughput, and the "calculate" button. The reason for facilitating two pull-down menus is to allow users to analyze network events between two linked nodes. After clicking the "calculate" button, depending on the statistic desired, one button or two buttons can appear below the "calculate" button. In the case depicted in Figure 3.6, the throughput option has only a "graph" button which can provide a throughput graph in a separate window.

The statistics tab can give users information needed to analyze the network at a higher level than other network visualizations. The statistics can be calculated using the parameters in the trace file. For example throughput is the rate at which a network sends receives data. It is rated in terms bits per second (bit/s). To find the throughput, we divide the packets received into the amount of forwarded packets over a certain time interval. Thus a good channel capacity of network connections can be found. Another good example is end-to-end delay which is the time taken for a packet to be transmitted across a network from source to destination. To find the end-to-end delay we subtract the packet sent time at source node from the packet receive time at the destination [37].



Location 400.0, 300.0
Time: 10.05 Sec

From : .... Aircraft Carrier
To : .... Satellite

Cluster Name : Cluster 1
Force Name : Air Force
Troop Name : . Troop 1
Node Name : ......Aircraft Carrier

Figure 3.6: A snapshot of prototype design showing the statistics tab including two pull-down menus, statistics and a throughput graph in a separated window

3.1.4 Other Features of the Prototype Design.

3.1.4.1 Zooming and Panning. The prototype design describes same zooming and panning facilities with the existing NetViz framework. The new NetViz design will also employ a mix of both manual and automated panning and zooming for navigating the space. Users can use panning by dragging the background of the display with the left mouse button down. When a new node is expanded, the display automatically pans, centering on the newly expanded network. Users can use manual-zooming by holding down the right mouse button moving the mouse up and down or using two icons on bottom of the right of the window. Simply clicking the right mouse button causes the display to automatically pan and zoom such that the entire visualized network fits within the display. The rendering components update to draw higher resolution photos when zoomed-in to double the normal scale.

3.1.4.2 Find and Location. Find field enables the user to quickly search a network item from visual presentation typing its name and clicking the "GO" button. Location tool shows selected item's X and Y coordinates. This information is included in defined tags -u (x-coordinate velocity), -v (y- coordinate velocity) in the trace file of NS-2. A field on the panel of the prototype design shows the location information. This information changes depending on where the user moves the cursor.

3.1.4.3 Control Panel. The control panel has the "start", "pause" and "stop" buttons. As mentioned in design specifications, these buttons give users the ability to control visualization through the simulation. Additionally, the time control area is presented to speed up or slow down the time to give the user the ability to manage the speed of visualization. File menu which is on the control panel currently contains one option, "Open a Trace File", which is used to manually open a trace file obtain from NS-2 simulator.

3.1.4.4 Summary Information about a Cluster. The left bottom side of the windows consists of a panel displaying a selected cluster's summary information. This panel includes the cluster's name, contents, neighbor clusters, and connection nodes. In the case illustrated in Figure 3.6, the cluster 1 is selected from the pull-down menu list. Users can see both the network events information on the view tab and summary information about a cluster on this panel.

3.1.4.5 Simulation status. The right bottom side of the windows consists of a field displaying the status of the connectivity. This field can show two statues which are "connected" and "not connected". Thus, users can be kept informed about the last status of the network and the network events as the visualization executes, as discussed in the design specifications.

3.2 Overview of NS-2 Result Processing

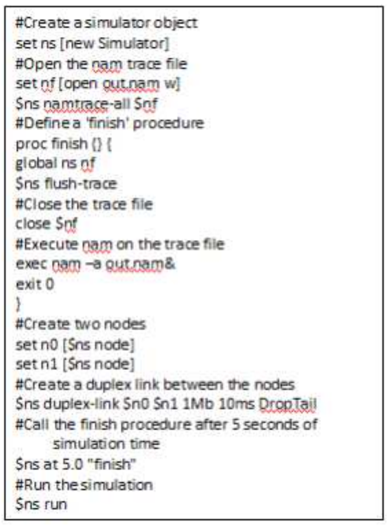
3.2.1 NS-2 OTcl File. As mentioned in Chapter 2, NS-2 is an object-oriented, discrete-event driven network simulator. It is an OTcl interpreter with network simulation object libraries written in C++ and OTcl. Researchers utilize network simulation scripts in Tcl (Tool Command Language) to create the network scenarios. Tcl is a scripting language which is used widely on embedded systems. Tcl scripts should be entered using a standard text editor and saved with the extension Tcl. An example of Tcl script is shown in Figure 3.7 (a). Since OTcl is Object-oriented extension of Tcl, the relationship between Tcl and OTcl is similar to C and C++ and all Tcl commands work on OTcl [38].

3.2.1.1 First OTcl Script for Analysis. A wired network is a network with physical cables connecting each system together. A simple OTcl script is shown in Figure 3.7(a) which creates 2 nodes (0, 1) and adds a duplex link which has 2 Mbps of bandwidth and 10 ms of delay between these two nodes [39]. It also connects the agents and run the simulation for 5 seconds. The topology of the scenario is depicted in Figure 3.7(b).

To run this script on a NS-2 installed machine, it's necessary to open a command console and type "ns scriptname.tcl" under the directory containing the Tcl script. Thus, NS-2 runs and we obtain a trace file which is generated according to the instructions included in the OTcl script.

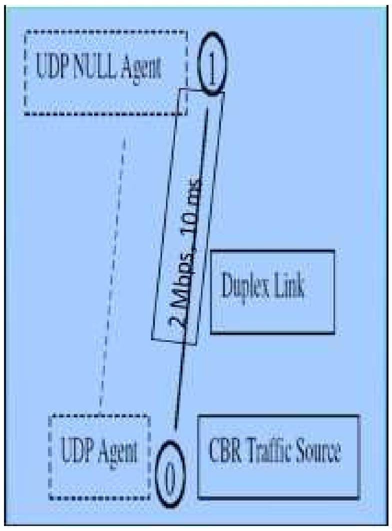
3.2.1.2 Second OTcl Script for Analysis. Transportation protocols are TCP, UDP, multicast which are the used in wired networks. Web, ftp, telnet, CBR, stochastic are the traffic sources used in wired networks. With the usage of TCP, UDP wired networks, ftp, CBR traffic sources, the OTcl script consisting of 4 nodes (n0, n1, n2, n3) is used to create a simple network configuration and run the simulation scenario in Figure 3.8.

In this wired network scenario consisting of 4 nodes, the duplex links between n0 and n2, and n1 and n2 have 2 Mbps of bandwidth and 10 ms of delay. The duplex link between n2 and n3 has 1.7 Mbps of bandwidth and 20 ms of delay. Each node uses a DropTail queue, of which the maximum size is 10. A TCP agent is attached to n0, and a connection is established to a TCP "sink" agent attached to n3. As default,



```
#Create a simulator object
set ns [new Simulator]
#Open the nam trace file
set nf [open out.nam w]
$ns namtrace-all $nf
#Define a 'finish' procedure
proc finish {} {
global ns nf
$ns flush-trace
#Close the trace file
close $nf
#Execute nam on the trace file
exec nam –a out.nam&
exit 0
}
#Create two nodes
set n0 [$ns node]
set n1 [$ns node]
#Create a duplex link between the nodes
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
#Call the finish procedure after 5 seconds of
      simulation time
$ns at 5.0 "finish"
#Run the simulation
$ns run
```

(a)



(b)

Figure 3.7: (a) A Simple Tcl script including 2 nodes. (b)The topology of this script showing two nodes and a duplex links which has 1 Mbps of bandwidth and 10 ms of delay between nodes

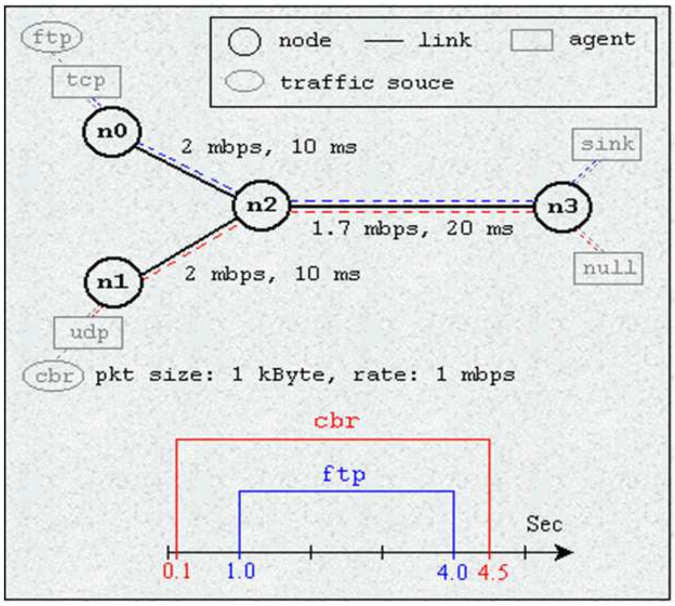


Figure 3.8: A simple network topology and simulation scenario consisting of 4 nodes, the duplex links between n0 and n2, and n1 and n2 have 2 Mbps of bandwidth and 10 ms of delay. The duplex link between n2 and n3 has 1.7 Mbps of bandwidth and 20 ms of delay [39].

the maximum size of a packet that a TCP agent can generate is 1KByte. A TCP "sink" agent generates and sends ACK packets to the sender (TCP agent) and frees the received packets. A UDP agent that is attached to n1 is connected to a "null" agent attached to n3. A "null" agent just frees the packets received. A "ftp" and a CBR traffic generator are attached to TCP and UDP agents respectively, and the CBR is configured to generate 1 KByte packets at the rate of 1 Mbps. The CBR is set to start at 0.1 sec and stop at 4.5 sec, and "ftp" is set to start at 1.0 sec and stop at 4.0 sec [39].

3.2.1.3 Joint Service Environment Scenario. Joint Service Environment (JSE) script is written to verify the new NetViz user interface. Since the new NetViz is capable of handling large networks, the JSE script is produced to meet demand for multiple network objects and events. For the purpose of our research, we aimed to generate this scenario including at least three clusters and twenty nodes during the scripting process. To emphasize the possible impact a network link going down, the scenario presents a large scale wired environment for monitoring a military network. Even though the JSE has nodes such as satellite and UAV, we assume that the JSE is a possible wired scenario consisting of 31 nodes and 4 clusters. As discussed in the Section 3.3, some modifications have to be done in JSE script file to provide visualization functions in NetViz. JSE scenario is utilized to illustrate the power of the new NetViz framework in Chapter IV.

3.2.2 Trace File Format. Figure 3.9 shows the flows of events for a Tcl file running in NS-2 to create a network scenario. As the simulation runs, trace files are generated. These trace files capture events occurring in the network and information which can be used in performance analysis such as the amount of packets transferred from source to destination, packet loss and, the delay in packets, etc.

As shown in Figure 4.10, in the trace file, each trace line starts with a packet event (+, -, d, r) descriptor followed by the other simulation events indicated with numbers. Definitions of NAM visualization packet events are shown in Figure 3.11.

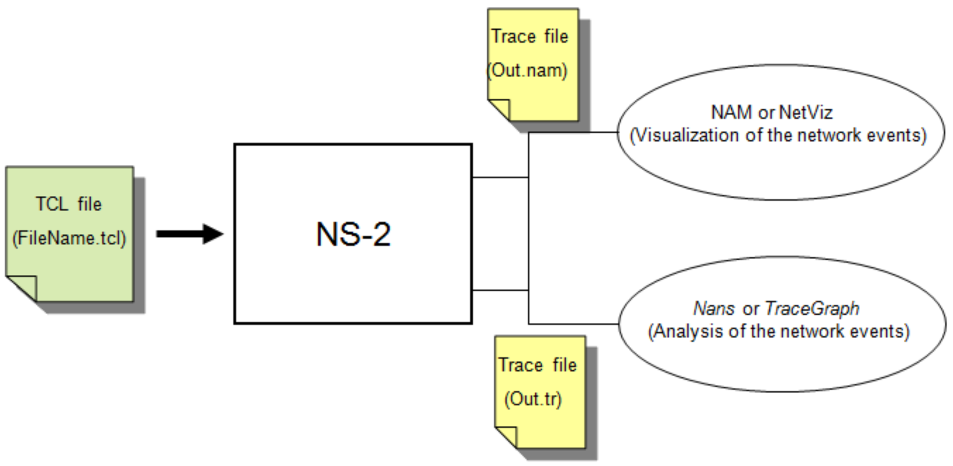
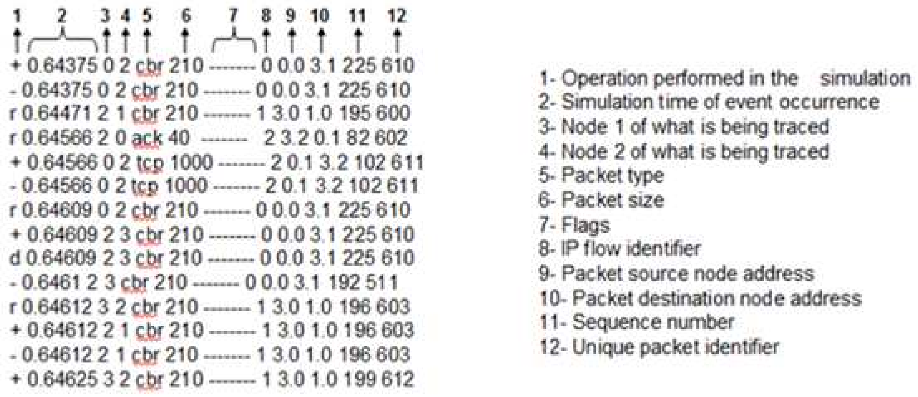


Figure 3.9: Flow of events for a NS-2 Tcl file.



```
1      2        3 4 5    6       7     8  9  10   11  12
+ 0.64375 0 2 cbr 210 ------- 0 0.0 3.1 225 610
- 0.64375 0 2 cbr 210 ------- 0 0.0 3.1 225 610
r 0.64471 2 1 cbr 210 ------- 1 3.0 1.0 195 600
r 0.64566 2 0 ack 40  -------  2 3.2 0.1 82 602
+ 0.64566 0 2 tcp 1000 ------- 2 0.1 3.2 102 611
- 0.64566 0 2 tcp 1000 ------- 2 0.1 3.2 102 611
r 0.64609 0 2 cbr 210 ------- 0 0.0 3.1 225 610
+ 0.64609 2 3 cbr 210 ------- 0 0.0 3.1 225 610
d 0.64609 2 3 cbr 210 ------- 0 0.0 3.1 225 610
- 0.6461 2 3 cbr 210 ------- 0 0.0 3.1 192 511
r 0.64612 3 2 cbr 210 ------- 1 3.0 1.0 196 603
+ 0.64612 2 1 cbr 210 ------- 1 3.0 1.0 196 603
- 0.64612 2 1 cbr 210 ------- 1 3.0 1.0 196 603
+ 0.64625 3 2 cbr 210 ------- 1 3.0 1.0 199 612
```

1- Operation performed in the simulation
2- Simulation time of event occurrence
3- Node 1 of what is being traced
4- Node 2 of what is being traced
5- Packet type
6- Packet size
7- Flags
8- IP flow identifier
9- Packet source node address
10- Packet destination node address
11- Sequence number
12- Unique packet identifier

Figure 3.10: Trace file format [4].

Nam Visualization Packet Events	
+	Packet is queued in a node
−	Packet is dequeued from a node
h	Packet begins transmission across a link
r	Packet is received by another node
d	Packet is dropped from node queue (queue overflow)

Figure 3.11: NAM Visualization Packet Events [4].

3.2.3 Simulation Visualization and Analysis. The trace files consists a huge amount of detail regarding protocol behavior, which are blocks of ASCII data and beyond the comprehending easily by the human brain. Researchers face challenges, including simultaneously displaying state in this large volume of data in the trace file, analyzing the scenario results, and characterizing dynamic interactions. Some forms of post processing techniques are needed to visualize and analyze the simulation results.

As depicted in Figure 3.9, running a script generates a NAM trace file (Out.nam) and a trace file called "Out.tr". These trace files can then be inspected using a visualization or analysis tool Out.nam can be used as an input to NAM or NetViz. Out.tr can be used with *nans* [11] or *tracegraph* [10] for the simulation analysis.

The trace file obtained from this script is used as an input for *nans* and *trace-graph* analyze tools to obtain needed graphs. Thus, it's expected to investigate ways to improve statistics tab of the new NetViz framework for the future work. Extracting the data set, *nans* and *tracegraph* calculate parameters such as one-way delay, throughput etc. and gives instantaneous results. The graphs are presented in the Section 3.3.3.2.

3.2.3.1 Analysis and Interaction Capabilities of NAM and NetViz. Visualization softwares operate similarly: process trace files generated during NS-2 execution, translate the data, and display the information to a monitor. NAM and NetViz visualization softwares discussed in Chapter 2 provide packet-level animation. They have limited analysis capabilities: NAM can graphically present information such as throughput and number of packets dropped at each link and NetViz provides node statistics such as packet receives, drops, and receive/drop percentages, when a user clicks on a desired node. However, additional features are needed to give the user the ability to perform network performance analysis at a higher level. For this purpose, the performance parameters that can be obtained through the trace files are plotted in the form of graphs using two analysis tools, *nans* and *tracegraph* in the following section.

NAM and NetViz have some user interface capabilities in common such as fast-forward/rewind, pause, jump slider. Other than scenario playback, the only user interaction with NAM is limited to setting up network scenarios. But it doesn't have the capability to let the user interact the system while a scenario is running. Similarly, one drawback of NetViz is being lack of capability of interacting with the simulator.

Under the guidance of the prototype design, it is expected to produce a new NetViz framework using *prefuse* toolkit. NetViz is coupled with the *mediator* tool to have the capability of rendering the simulation during NS-2 execution. It's also expected to strengthen the framework with visualization functions. In short, this framework has two big differences from NAM and the old version of NetViz. The first difference is the new NetViz framework provides user interaction with NS-2 using the *mediator* tool. Second, the way for data presentation of large network scenarios using visualization functions discussed in this Chapter. These differences are detailed in Chapter IV.

3.2.3.2 Analysis of the Simple Wired Network Scenario. The trace analyzer softwares such as *nans* and *tracegraph* meant to ease the task of analyzing network performance after network simulation on NS-2. Since visualization softwares provide limited help for analyzing and understanding the data in trace file, *nans* and *tracegraph* can be used as observation tools for the post simulation processes of NS-2 trace files. These tools are capable of calculating many parameters included in the trace file characterizing network simulation. Obtaining visual displays (graphs) from these tools, users can understand the large amount of data easily.

Usage of these post processing tools provides a useful foresight which would help the progress of NetViz framework giving a potential to plug these graphs into the user interface. Implementation of this new analysis capability changes depending on *prefuse* toolkits feasibility.

The graphs give users control and information needed to analyze the network at a higher level than other network visualizations. A number of performance parameters can be obtained from the trace file. These parameters can be used to calculate throughput, end-to-end delay or Round Trip Time (RTT).

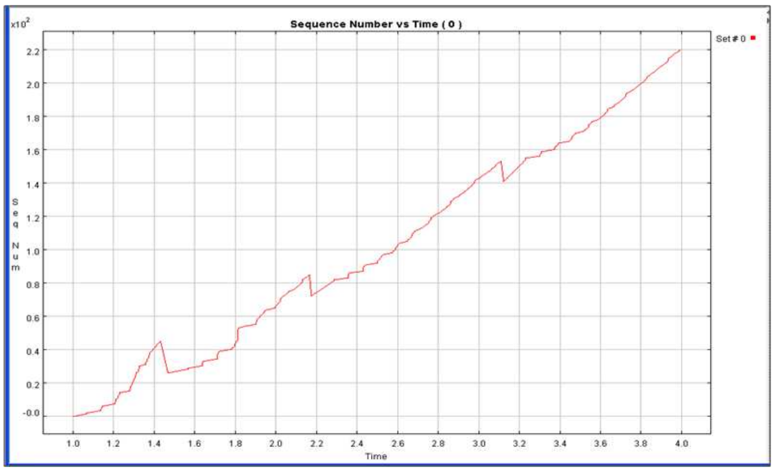


Figure 3.12: A snapshot of *nans* observation tool result showing sequence number versus time (0).

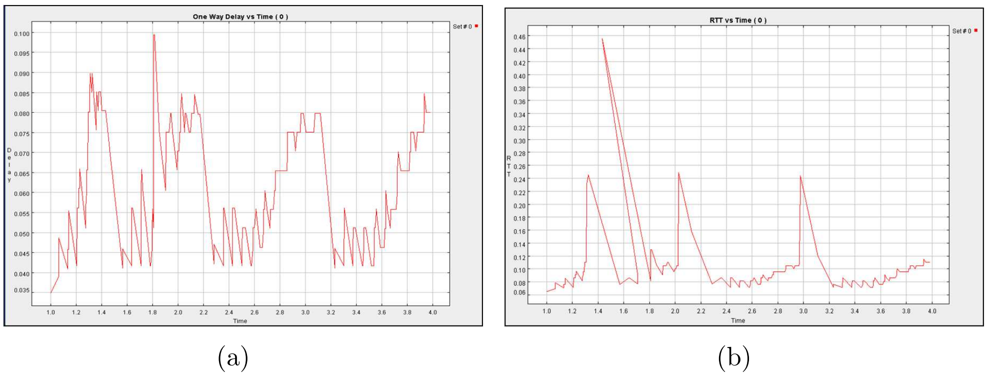


Figure 3.13: (a) *Nans* 2D graph showing one way delay versus time (b) RTT versus time

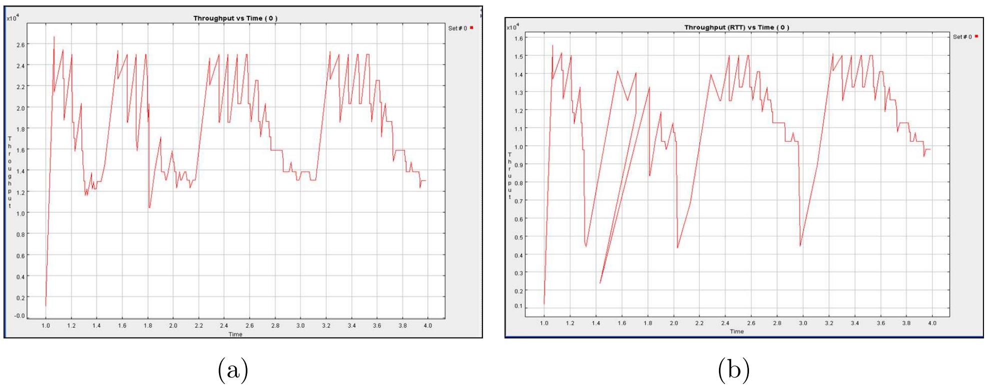
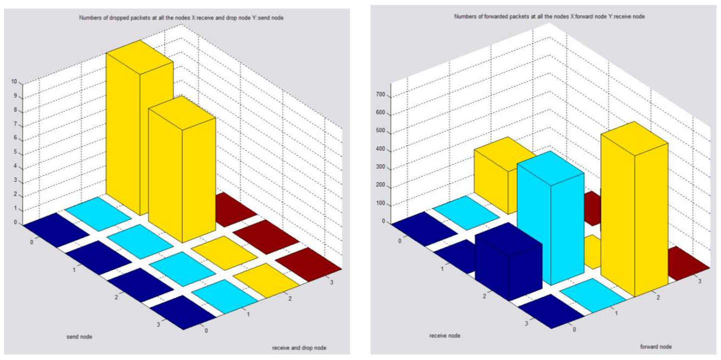
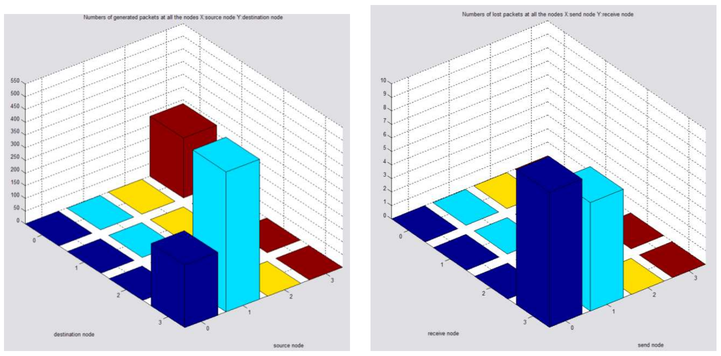


Figure 3.14: (a) *Nans* Throughput versus time (b) Throughput (RTT) versus time



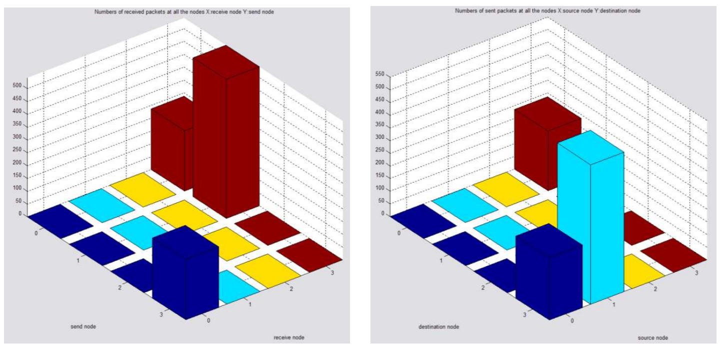
(a) (b)

Figure 3.15: (a) *Tracegraph* 3D graph of nodes showing numbers of dropped packets at all nodes (b) Numbers of forwarded packets at all nodes.



(a) (b)

Figure 3.16: (a) Numbers of generated packets at all nodes (b) Numbers of forwarded packets at all nodes.



(a) (b)

Figure 3.17: (a) Number of received packets at all nodes (b) Numbers of sent packets at all nodes.

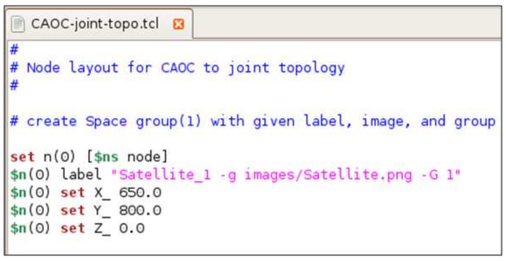
The trace file that obtained from the second OTcl script described in the previous section is analyzed using *nans* and *tracegraph*. *Nans* is used to obtain 2D graphs such as Sequence number /throughput/end to end delay versus time. These graphs are shown in Figures 3.12, 13 and 14. *Tracegraph* is used to obtain 3D graphs such as numbers of forwarded/ dropped/received packets shown on source and destination node sides. Showing different perspectives of these graphs, *tracegraph* allows us to detail the observation of network events. These graphs are shown in Figures 3.15, 3.16 and 3.17.

3.3 Implementation Process

This research expanded the network visualization framework for the dynamic interactions by adding object interaction functionality to the actual classes. The *prefuse* polylithic design is able to emanate logic across different classes [9]. The multiple class approach incorporated in the polylithic design of *prefuse* is appropriate for the development of the new NetViz framework. Because the polylithic class hierarchy of data structures enables the implementation of behaviors for the main visual objects such as nodes, links, and packets on the screen. Adding the node, link, and packet functionality to the existing *prefuse* classes does not require additional implementation.

In our research, an interaction between NetViz and NS-2 is provided using the *mediator* tool. The *mediator* tool provides TCP/IP connection service for NS-2 and NetViz. The interaction is accomplished by changing the software of both NetViz and NS-2. [12] produced the *mediator* tool and provided modified NS-2 code. Our research adds the **StreamingSocketController** class to the existing NetViz classes to connect NetViz to *mediator* . Additionally, some modifications have to be done in the NS-2 script files to provide visualization functions.

The prototype design of the user interface presented in the previous section includes multiple network visualization functions such as clustering, labeling and color coding. To implement these functions in the new NetViz framework, some of the NS-2 script files are modified. The reason for this is that NS-2 does not originally support some of the properties needed to display such visualization functions as clustering and color coding. We modify *topo.tcl* file which is set as a topology file in the Tcl file of JSE scenario. The node information in this file is enhanced adding new tags such as label, image, force and cluster for the visualization functions. The second line in Figure 3.18 (a) is separated into three lines using quotation marks for each word indicating label, image and cluster (See Figure 3.18 (b)). Then a new line is added that includes one of the force names such as “army”, “navy”, “air force”. By repeating this procedure for each node, a new Tcl file obtained.

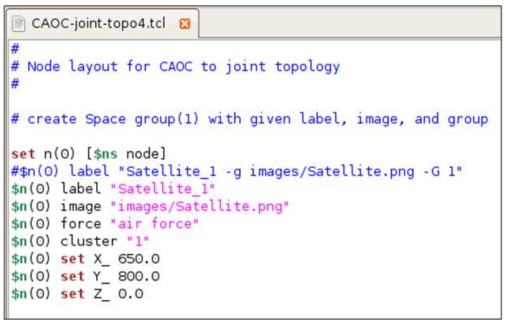


```
CAOC-joint-topo.tcl
#
# Node layout for CAOC to joint topology
#

# create Space group(1) with given label, image, and group

set n(0) [$ns node]
$n(0) label "Satellite_1 -g images/Satellite.png -G 1"
$n(0) set X_ 650.0
$n(0) set Y_ 800.0
$n(0) set Z_ 0.0
```

(a) The original *JSE-joint-topo.tcl* file



```
CAOC-joint-topo4.tcl
#
# Node layout for CAOC to joint topology
#

# create Space group(1) with given label, image, and group

set n(0) [$ns node]
#$n(0) label "Satellite_1 -g images/Satellite.png -G 1"
$n(0) label "Satellite_1"
$n(0) image "images/Satellite.png"
$n(0) force "air force"
$n(0) cluster "1"
$n(0) set X_ 650.0
$n(0) set Y_ 800.0
$n(0) set Z_ 0.0
```

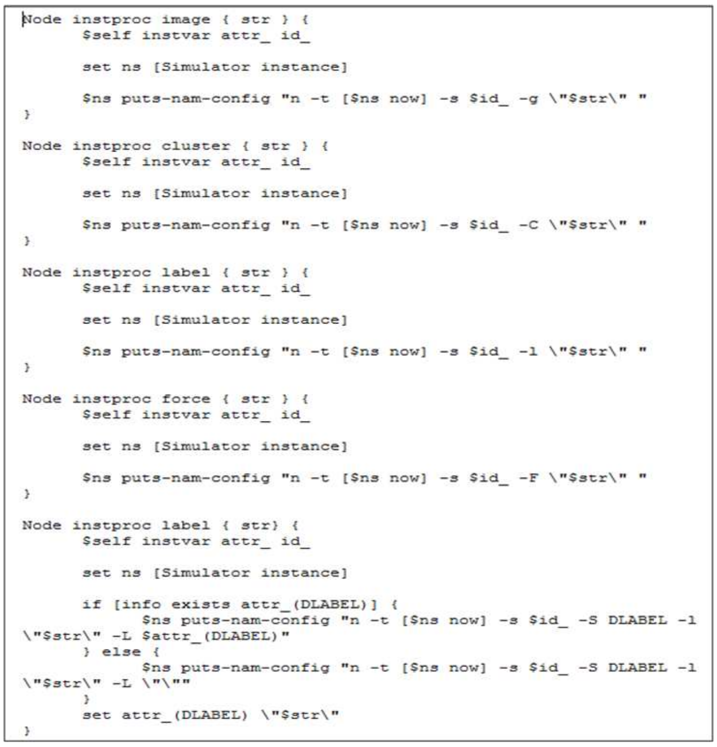
(b) Modified *JSE-joint-topo.tcl* file including label, image, force and cluster tags. The second line of the original file is separated into 3 lines and then one more line is added

Figure 3.18: Modifications in *JSE-joint-topo.tcl* file.

Thus, the node information is not received all at once, but is instead received piece by piece in the new Tcl file. The **NodeParser** class is changed and made flexible enough to allow node variable initialization across multiple trace lines and add support for the image, label, force, cluster tags. After the change, the **NodeParser** class is capable of receiving a node’s label in one line and then its cluster a few lines later.

The modifications in topo.tcl file created a need for change in one of the NS-2 file. A number of NAM keywords are added to *ns-namsupp.tcl* file to support the modifications topo.tcl file. The keywords shown in Figure 3.19 are added into *ns-*

namsupp.tcl to facilitate clustering, labeling and color coding on the new NetViz framework.



```
Node instproc image { str } {
        $self instvar attr_ id_

        set ns [Simulator instance]

        $ns puts-nam-config "n -t [$ns now] -s $id_ -g \"$str\" "
}

Node instproc cluster { str } {
        $self instvar attr_ id_

        set ns [Simulator instance]

        $ns puts-nam-config "n -t [$ns now] -s $id_ -C \"$str\" "
}

Node instproc label { str } {
        $self instvar attr_ id_

        set ns [Simulator instance]

        $ns puts-nam-config "n -t [$ns now] -s $id_ -l \"$str\" "
}

Node instproc force { str } {
        $self instvar attr_ id_

        set ns [Simulator instance]

        $ns puts-nam-config "n -t [$ns now] -s $id_ -F \"$str\" "
}

Node instproc label { str} {
        $self instvar attr_ id_

        set ns [Simulator instance]

        if [info exists attr_(DLABEL)] {
                $ns puts-nam-config "n -t [$ns now] -s $id_ -S DLABEL -l
\"$str\" -L $attr_(DLABEL)"
        } else {
                $ns puts-nam-config "n -t [$ns now] -s $id_ -S DLABEL -l
\"$str\" -L \"\""
        }
        set attr_(DLABEL) \"$str\"
}
```

Figure 3.19: Additional keywords in *ns-namsupp.tcl* to facilitate clustering, labeling, filtering and color coding functions.

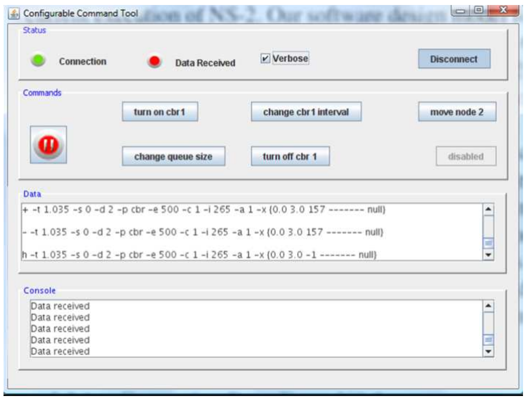
We kept most of the classes same such as **GroupedLayout** which was added to the existing *prefuse* classes in [5]. To allow further flexibility in the view, **nvReneder-erFactory** is modified to allow the user to customize what data is displayed and how it is presented. **ComplexLabelRenderer** is changed to support the modifications to **nvRendererFactory**. The **ClusterDrawActions**, **StreamingTraceReader**, **StateController** classes are also added to the existing *prefuse* classes.

3.4 Visualization/Simulator Interaction Using Mediator Tool

The *mediator* tool [12] was produced to establish a communication link with an external simulator passing new trace information to the visualization parser through the link. Our research used the *mediator* tool to give NetViz users the ability to interact with NS-2 as it executes. For the purpose of our research, NetViz is integrated with the *mediator* and modified NS-2 code to establish a complete simulation execution with rendering the visualization and providing command feedback.

As described in the previous section, the new NetViz framework does not use static data unlike NAM and the old version of Netviz. The reason is that the visualization-simulation interaction could be established streaming data from a running the Tcl file on NS-2 code. Thus, data is added and modified dynamically as the trace file is streamed and scenario changed.

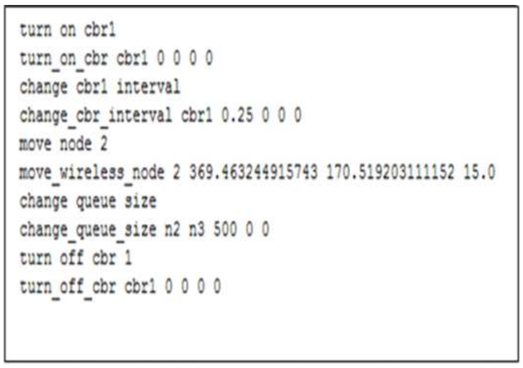
To run the new NetViz framework integrated with the *mediator* and modified NS-2 code, we first run the *mediator* tool. Then we set Configurable Command Tool (CCT) [12] which requires an ip address and port number of the *mediator* computer (see Figure 3.20(a)). After setting the CCT for the local computer these steps are followed: (1) Run NetViz screen streaming (2) Run the Tcl file on the modified NS-2 (3) Click on "Start" button on the NetViz user interface.



```
+ -t 1.035 -s 0 -d 2 -p cbr -e 500 -c 1 -i 265 -a 1 -x {0.0 3.0 157 ------- null}
- -t 1.035 -s 0 -d 2 -p cbr -e 500 -c 1 -i 265 -a 1 -x {0.0 3.0 157 ------- null}
h -t 1.035 -s 0 -d 2 -p cbr -e 500 -c 1 -i 265 -a 1 -x {0.0 3.0 -1 ------- null}
```

```
Data received
Data received
Data received
Data received
Data received
```

(a) Configurable command tool GUI depicting status, commands, data, and console panels. The Connection indicator turns green when a connection is established with the mediator by pushing the Connect button.

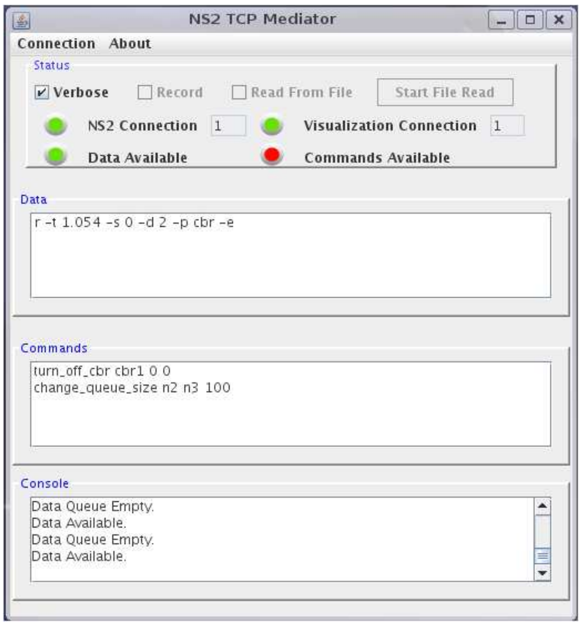


```
turn on cbr1
turn_on_cbr cbr1 0 0 0 0
change cbr1 interval
change_cbr_interval cbr1 0.25 0 0 0
move node 2
move_wireless_node 2 369.463244915743 170.519203111152 15.0
change queue size
change_queue_size n2 n3 500 0 0
turn off cbr 1
turn_off_cbr cbr1 0 0 0 0
```

(b) The configuration file including the needed commands.

Figure 3.20: Configurable command tool GUI and the configuration file.

Figure 3.21 shows the *mediator* GUI that we see when we first start this tool. Running Tcl file on the modified NS-2 requires to run "./ns scenarioName.tcl" on the command console. Clicking on "Start" button on the NetViz user interface will start the visualization. Thus, users will have the ability to pause and resume the visualization any time dynamically interacting with NS-2. There is an offset time to make sure NS-2 stays ahead of NetViz. NetViz has to be a little behind from NS-2, because sending data to NetViz takes time. There is a configuration file including offset time information which is set as 1.1 second. This file can be changed depending on the system needs.



```
r -t 1.054 -s 0 -d 2 -p cbr -e
```

```
turn_off_cbr cbr1 0 0
change_queue_size n2 n3 100
```

```
Data Queue Empty.
Data Available.
Data Queue Empty.
Data Available.
```

Figure 3.21: Mediator GUI depicting status, data commands an console panel. The four indicators turn green when connection or information is available [12].

Currently, the only command that works on the command tab is changing "queue size". The command tab functionality for other commands is proceeding to provide direct manipulation of the scenario. Other than command tab, the additional panel, CCT, including commands such as "turn ON/OFF cbr" and "change the interval" can be used to provide the direct manipulation of simulation. At the time this research was written, CCT panel has been successfully coupled with the NetViz and now has the ability to manipulate the scenario. The command buttons on the CCT panel are configurable by a configuration file including needed command words as shown in Figure 3.20(b).

3.5 Demonstration

Providing TCP/IP connection service for NS-2 and NetViz clients, the *mediator* allows multiple visualizations to connect to NS-2. Taking advantages of the interaction capability of the new NetViz, a simulation execution scenario can be built integrating NetViz clients with a *mediator*, and a modified NS-2 code. In the following sections, first we demonstrate two NetViz clients linked to NS-2. Then, we purpose to increase numbers of clients to six adding four more clients to ensure that the scenario works on the different operating systems too. Our main goal is to demonstrate that the display of each NetViz client can be customized differently by the users.

3.5.1 Linking Two NetViz Clients to NS-2. In Figure 3.22, a simulation execution scenario is demonstrated integrating two NetViz clients with the *mediator* running on windows machines, and the modified NS-2 code running on linux machine. Before running this application, the *mediator* computer must be configured in the topology file of NS-2. The CCT also must be set for each NetViz client typing required ip address and port number of the *mediator* computer. In this scenario, the order of running programs as following: (1) Run the *mediator*, (2) Run the CCT script for each computer (typing the ip address and port number of the mediator computer), (3) Run NetViz screen streaming (4) Run NS-2 (5) Click on "Start" button on the NetViz user interface.

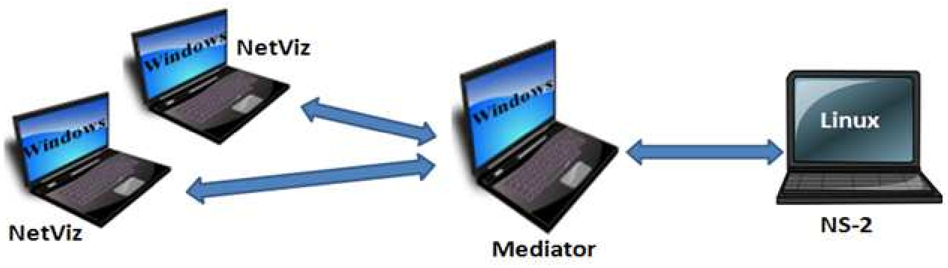


Figure 3.22: An illustration of the execution scenario which is composed of NS-2 running on linux operating system, the *mediator*, and two NetViz clients running on windows operating system.

The interaction between the elements of this execution scenario can be accomplished making needed changes described in Section 3.3. The *mediator* tool provides TCP/IP connection service for NS-2 and NetViz. Thus, it allows the NetViz computers to stream data from NS-2 connecting to it. In the case depicted in Figure 3.22, the flow of events between NS-2 (server) and NetViz computers (clients) are shown. The center of the application is the *mediator* collecting commands from each connected NetViz and passing them to NS-2. Then it receives data from NS-2 and duplicates it for each NetViz client. This will allow multiple clients to see the same simulation at the same time and make their own customizations on the display.

3.5.2 Distributed Execution Scenario with Six NetViz Clients. A distributed execution scenario can be obtained adding the previous scenario four more NetViz clients. The *mediator* runs on one of the NetViz computer. The CCT can be set for each computer typing ip address and port number of the mediator computer. Thus, the distributed system can be run following the same order of running programs defined in the previous section. Figure 3.23 shows the illustration of the execution scenario which is composed of NS-2 running on linux operating system, the *mediator* and five NetViz clients running on windows operating system and a NetViz client running on Mac operating system.

Needed changes described in Section 3.3 should be done before the execution starts. It's expected to accomplish the interaction between the elements of the scenario to prove that multiple NetViz clients running on different operating systems can be linked to NS-2. Thus users of each NetViz client can change visualization parameters differently on the screen. In the application depicted in Figure 3.23, one of the NetViz clients and the *mediator* are run on the same windows computer. The *mediator* collects commands from each connected Netviz clients and passes them to NS-2. Then it receives data from NS-2 and multiplies it for six NetViz clients. Each NetViz client can see the same simulation and use different visualization parameters on the user interface.

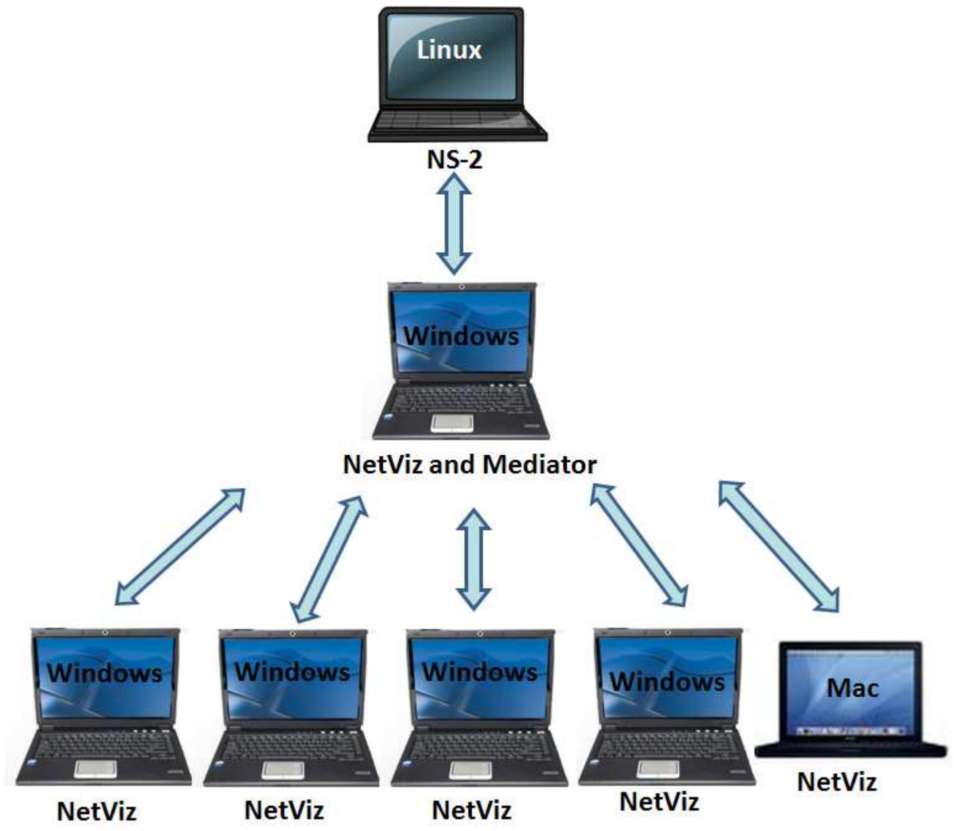


Figure 3.23: An illustration of the execution scenario which is composed of NS-2 running on linux operating system, the mediator, five NetViz clients running on windows operating system and one NetViz client running on Mac operating system.

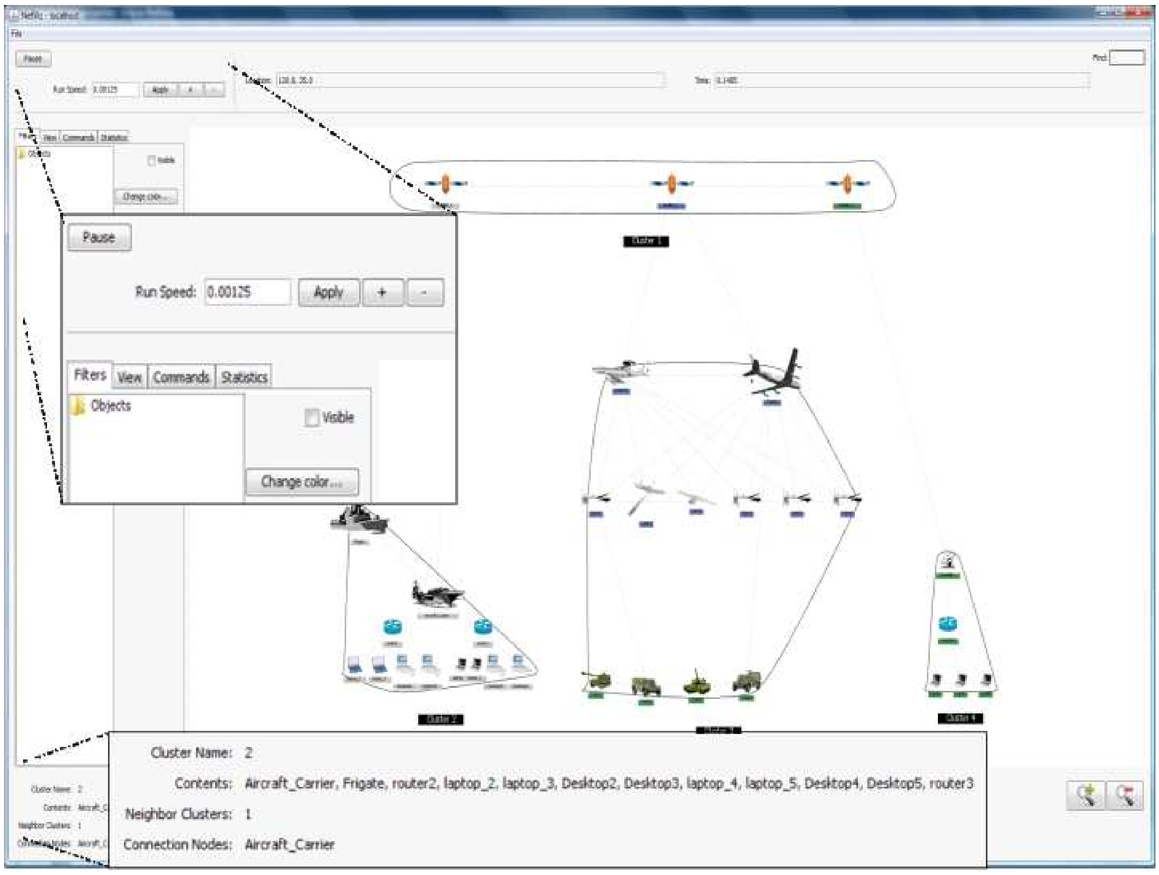
IV. The Test and Validation of the New NetViz Framework Design

This research designed and implemented a new NetViz framework with dynamic interactions for observing network events in large-scale network scenarios using the *prefuse* toolkit. Before constructing the new NetViz framework, a prototype design was needed to determine a route for implementing interactive techniques and the insight obtained from researched studies. Under guidance of this prototype design, our research produced the framework with the capability of meeting user needs as mentioned in design specifications described in Chapter III. The design builds upon same node-link network layout and the parser architecture defined in [5] to contribute customized network visualization functions. The visualization functions such as clustering, labeling, filtering and color coding assist in accessing network objects and events. These functions support four tabs consisting of buttons, menus, and sliders which provide control over the network objects for different tasks. The final framework design has tolerable differences from the prototype design because of the limits of NS-2 and the Java library. These differences are discussed while describing the distinct features of the user interface in the following section.

This research also tested a number of visualization scenarios by visual inspection. The main scenario detailed in this research is the JSE scenario because it presents a large network topology. JSE scenario is visualized for testing distinct features of the new NetViz framework. In addition to JSE scenario, a simple wired scenario is visualized to verify the interaction between the new NetViz and NS-2. The JSE scenario is visualized in both NAM and the old version of NetViz framework comparing the resultant visualizations with the same scenario visualized by the new version of the NetViz framework.

4.1 Testing Distinct Features of the New NetViz User Interface

The framework empowers the visualization with wide variety of new and enhanced features including clustering, filtering, labeling, selecting nodes and color coding. These new features are complemented by adding effective interactive techniques to the user interface giving users the ability to reduce the complexity of networks. Users can focus on desired areas of the large networks or change the views of objects by utilizing multiple options provided by the comprehensive user interface. This section establishes dynamic interactions accurately by visualizing JSE scenario for each feature of this user interface. The features of the user interface are described in terms of their purposes and the differences from the prototype design. Figure 3.1 shows visualization functions discussed in Chapter III assist in accessing network objects and events, supporting four tabs (filters, view, command and statistics) consisting of buttons, menus, and sliders. A closer view is shown in Figure 3.1. All the design steps in the prototype design could not be implemented because of the limits of NS-2 and the Java library as discussed in Chapter V.



Cluster Name: 2
Contents: Aircraft_Carrier, Frigate, router2, laptop_2, laptop_3, Desktop2, Desktop3, laptop_4, laptop_5, Desktop4, Desktop5, router3
Neighbor Clusters: 1
Connection Nodes: Aircraft_Carrier

Figure 4.1: This framework's visualization of JSE scenario displaying four clusters which include multiple nodes.

4.1.1 Clustering. This research investigates the use of clustering algorithms as a solution for improving the efficiency of large network scenarios. Clustering is aimed to accomplish filtering function in the prototype design described in Chapter III. The prototype design shows the semi-transparency for three clusters depicted in Figure 3.1. The reason for designing semi-transparency was to enable users to identify the images and labels of nodes indicating clusters in transparent colors. Thus, the users would concentrate on a particular network object and apply the filter table settings for the other members of the cluster (See figure 3.1). But the Java swing library limited the implementation of the filters table to the framework as discussed in the next section. However, different settings for filtering which serve to the same purpose and the semi-transparency are implemented to the new NetViz framework supporting the assignment of nodes to multiple clusters.

As shown in Figure 4.1, clusters are built to identify group structures based on the linkage of nodes in the network. Users should be able to view the communication behavior of these nodes as mentioned in section 3.1.1. For this reason, the clustering is used to ensure useful topology based groupings fast enough to support real-time interaction which is provided utilizing the *mediator* tool in our research.

In our application, the algorithm of clustering first presents each node in its own cluster, and then it minimizes the cluster's borders as users set the filters (See Figure 4.4). The tree view on the filters tab (left hand side of the screen) is used for this purpose. A closer view of the tree view is shown in Figure 4.2. Clicking on the "Objects" folder, users can obtain this tree view and see all the components to determine which item would be removed from the clusters.

In Figure 4.1, the left bottom side of the window consists of a panel displaying a selected cluster's summary information just like designed in Chapter III. The panel includes the cluster's name, contents, neighbor clusters, and connection nodes. This provides an insight about the selected cluster by means of purposed task.

Another unique property of the clustering algorithm is to give the users the ability to customize the cluster colors. As shown in Figure 4.2, with the click of "Change Color" button on the filters tab, a separated window showing various colors appears. This window lets users customize the network environment choosing a variety of transparent colors for the clusters.

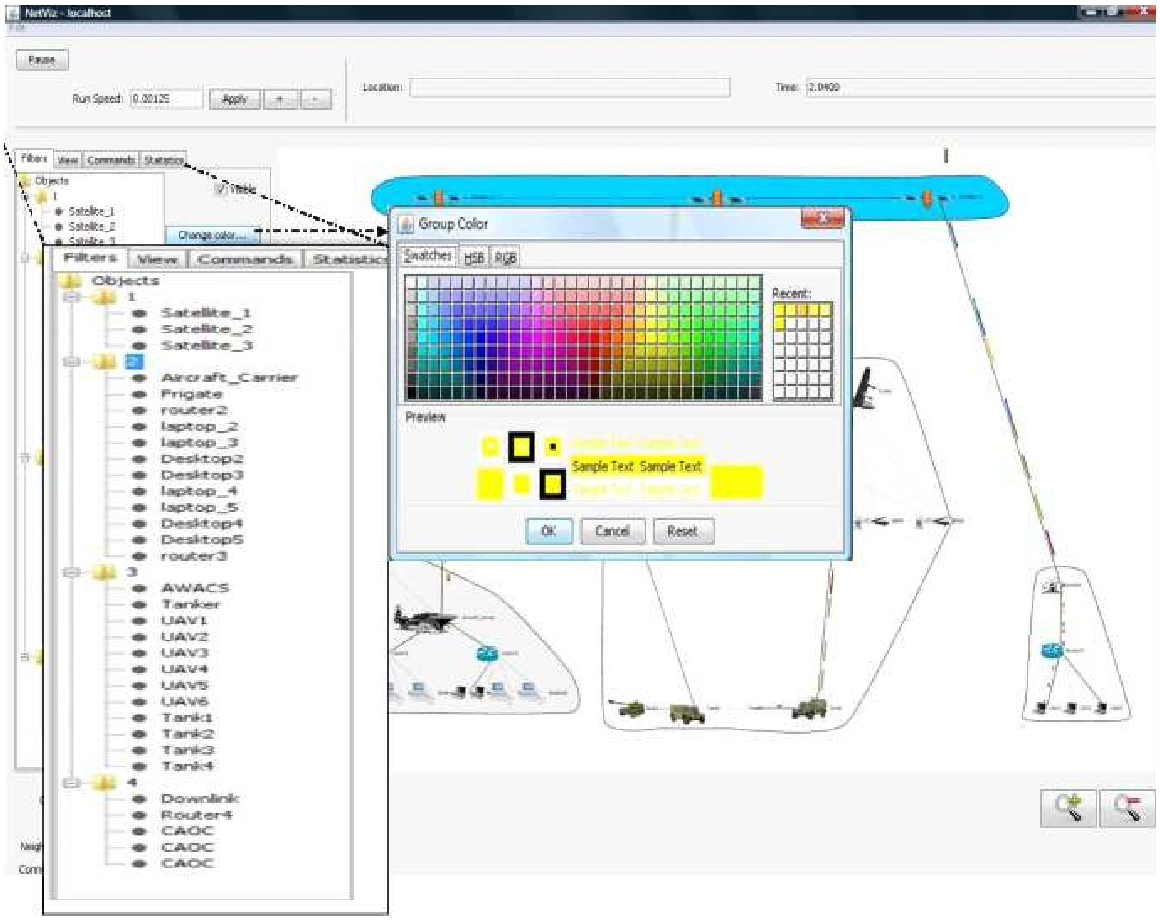


Figure 4.2: The screenshot of the new NetViz framework showing filters tab and "group color" panel which enables users to customize the colors of the clusters.

4.1.2 Filtering. Filtering data allows the user the ability to limit which data are displayed on the visual presentation. In the prototype design, the rows of filter settings table were planned to refer to clusters, which has two states, "Y" or "N" (See Section 3.1.3.1). The specific objective was to keep informed users allowing them to set the visibility of network objects displayed on a filter settings table. Because the

Java swing tables are not flexible enough to implement filters settings table, we could not implement this table, as mentioned in Chapter V. Instead of filter settings table a small visibility check-box is used to set the network objects invisible or visible on the right hand side of the filters tab. The hierarchical order display described in the prototype design could not be implemented either.

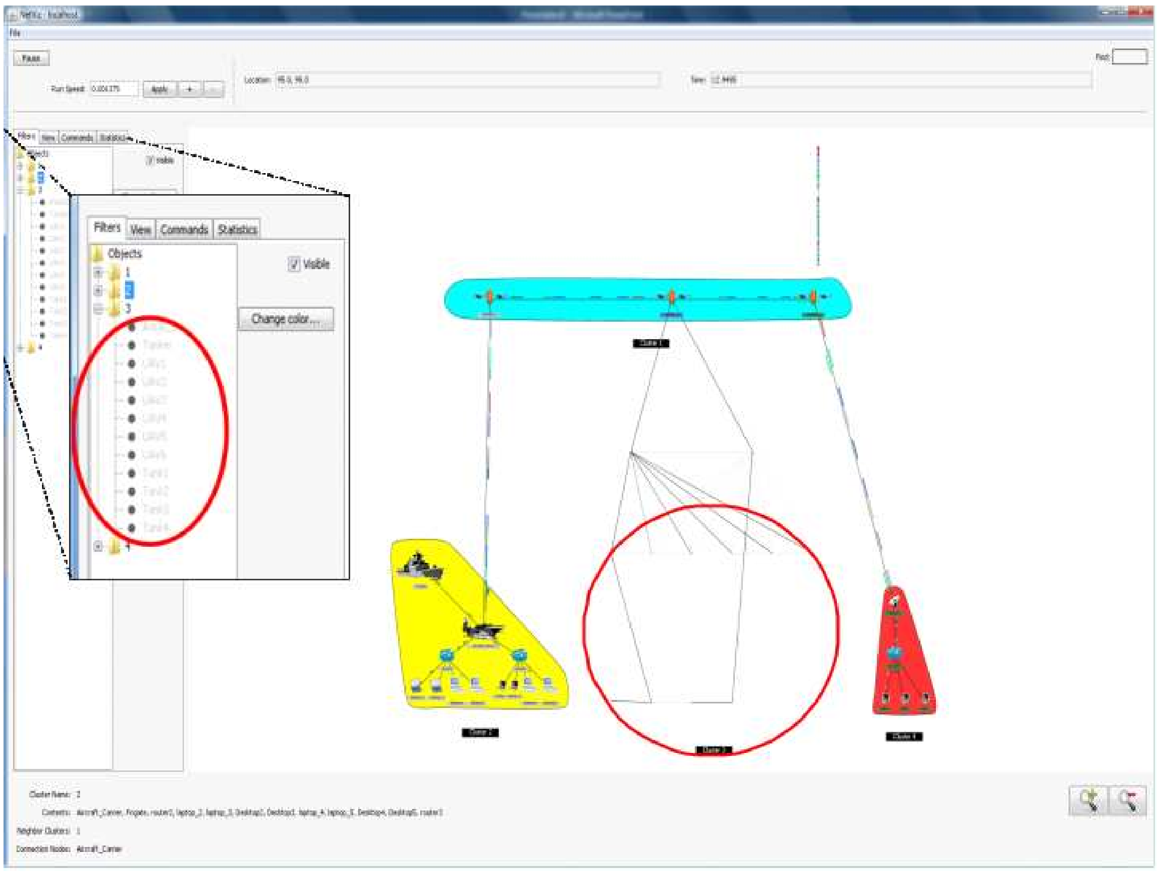
In our application, the clustering network visualization function is used to accomplish filtering function. The clusters consisting of nodes are organized in the tree structure. The tree view can be expanded or collapsed by clicking on the cluster folders. In the tree representation of the clusters, to remove a cluster or node from the display, users can click on the desired item and then deactivate the visible check-box.

As shown in Figure 4.3 and 4.4, other than the hierarchical relations of nodes, NetViz filters tab shows a tree representation of the clusters consisting of nodes on the left hand side. By deactivating the visible check-box after clicking on the "Cluster 3" on the tree view, the "Cluster 3" is removed from the display of Figure 4.3. Thus, all the objects included in the "Cluster 3" are made invisible but links.

In the same way, the "Frigate" node is removed from the display of Figure 4.4. Removing this node from the display changes the visible cluster structure reducing the borders of the cluster 2. By activating the visible check-box after clicking on the "Frigate" node, users can revert the display to the initial state of the clustering.

Removing the "Cluster 3" and the "Frigate" from the Figure 4.3 and 4.4 displays caused these items to be desaturated in the tree view, reverting names of them to grayscale. This provides awareness on the tree view letting users know which item is removed. Circles on both figures indicate the names in gray and the location of removed items. Thus, configuring the visual representation by the filters tab, the complexity of the display can be reduced in the direction of user needs.

4.1.3 Labeling. Knowing that the use of imagery is not enough to establish a logical relationship among nodes, the names of military components are displayed on the prototype design screen in Chapter III for ease of interpretation. The goal was



Run Speed: 0.004375

Location: 95.0, 95.0

Time: 12.9495

Cluster Name: 2

Contents: Aircraft_Carrier, Frigate, router0, laptop_2, laptop_3, Desktop2, Desktop3, laptop_4, laptop_5, Desktop4, Desktop5, router1

Neighbor Clusters: 1

Connection Nodes: Aircraft_Carrier

Figure 4.3: This framework's visualization of JSE scenario displaying 3 clusters. By clicking on "Cluster 3" on the tree view of filters tab and deactivating the visible check-box, "Cluster 3" is removed from the display.

basically to enable users to identify nodes using labeling function in the design. In our application, the labeling function is also used as a basis for the functionality of filtering, color coding, find field and selecting node. Establishing labeling function is needed to accomplish these functions. These functions described as distinct features of the new NetViz framework in this chapter.

To implement the labeling function, the compatibility of the *prefuse* classes with NS-2 is examined in Chapter III. Adding a number of names and keywords into the some of the NS-2 files, labeling is facilitated in the new NetViz framework as described in Section 3.3. Even though military components are commonly referred by various names in real world scenario information, general names are added into the

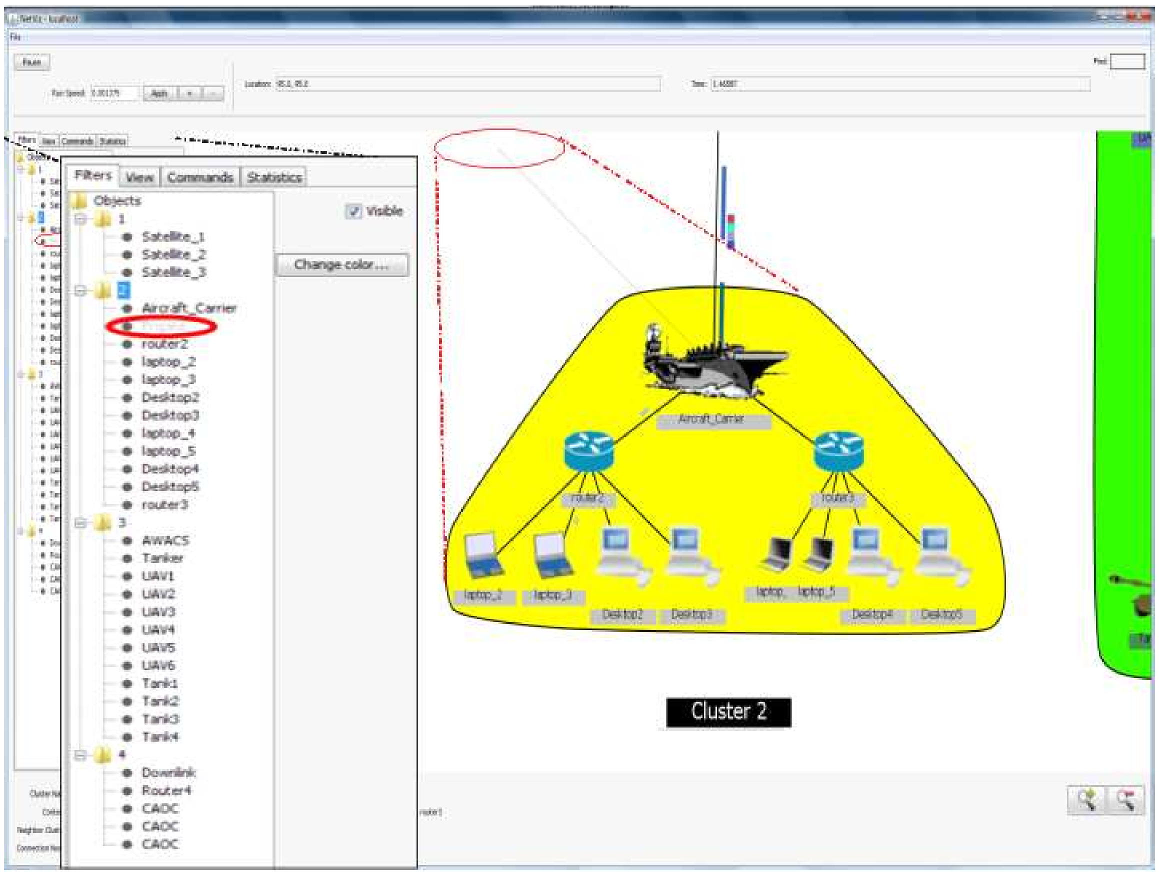
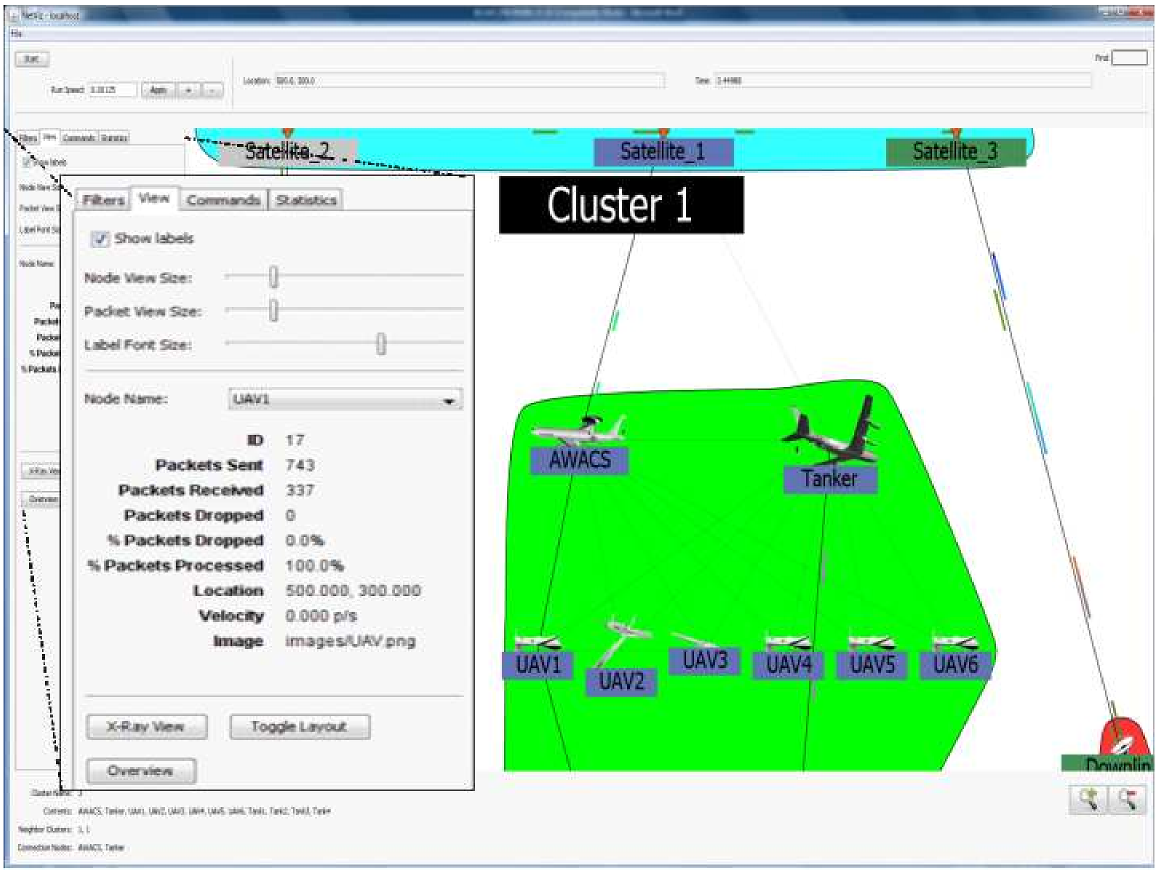


Figure 4.4: This framework's visualization of JSE scenario displaying cluster 2. Circles indicate that "Frigate" node is removed from the display by clicking on this node on the tree view and deactivating the visible check-box on filters tab. Removing this node from the display changes the visible cluster structure reducing the borders of the cluster 2.

NS-2 topology file for the JSE scenario. In the Figure 4.4, the visible cluster structure of "Cluster 2" is changed removing "Frigate" image and label from the display.

Various configurations can be composed using the three sliders on the view tab as designed in Chapter III. Figure 4.5 shows some components of cluster 1 and 3. The cursor of "label font size" slider is dragged to the right. Thus, labels grow slightly larger to increase awareness of the names of nodes and color of the force sides.

"Labels ON/OFF" was designed in Chapter III to decrease of complexity. This trade off refers to "show labels" check-box on the view tab of the new NetViz. Circle in Figure 4.6 indicates this check-box which is deactivated to remove all labels from the display. "Show labels" check-box can be used when "Zoom in" performed by users



☑ Show labels

Node View Size:

Packet View Size:

Label Font Size:

Node Name: UAV1

ID	17
Packets Sent	743
Packets Received	337
Packets Dropped	0
% Packets Dropped	0.0%
% Packets Processed	100.0%
Location	500.000, 300.000
Velocity	0.000 p/s
Image	images/UAV.png

Figure 4.5: View tab of the new NetViz user interface. The cursor of "label font size" slider is dragged to the right to make font size of labels larger to recognize them clearly. Zooming and panning performed to display a part of cluster 1 and 3.

to concentrate on a specified area of the network. While this will dilute the display, it might break the logical completeness of the network. Same view can be obtained by dragging the cursor of "label view size" slider to the left.

4.1.4 Color Coding. This research used color to encode both topological and non-topological properties of the forces and the clusters. A color-coded filter settings table was described in the prototype design in Chapter III. The hierarchical order next to this table was also color-coded to indicate the forces. The Java swing library limited us implementing the filter settings table and the hierarchical order. In our application, the tree view shown on the left side of the Figure 4.4 is produced, which

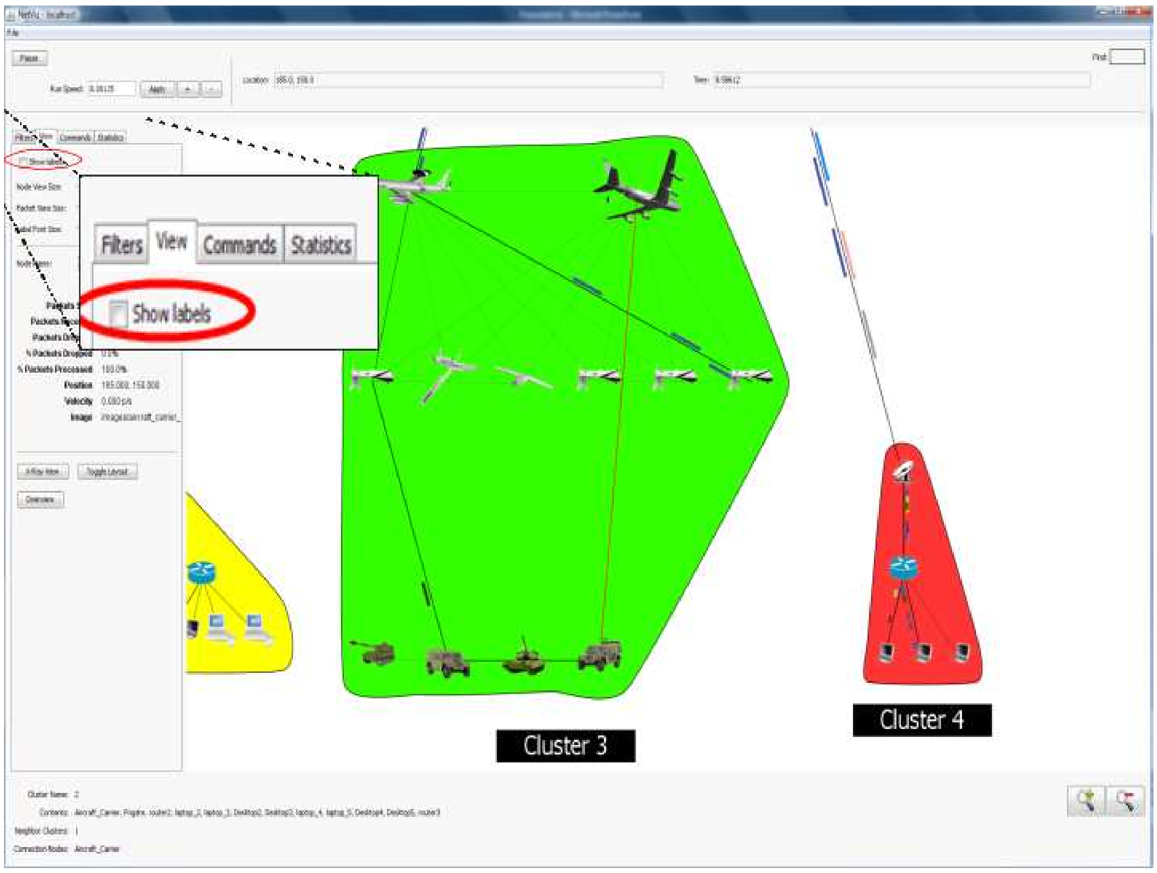
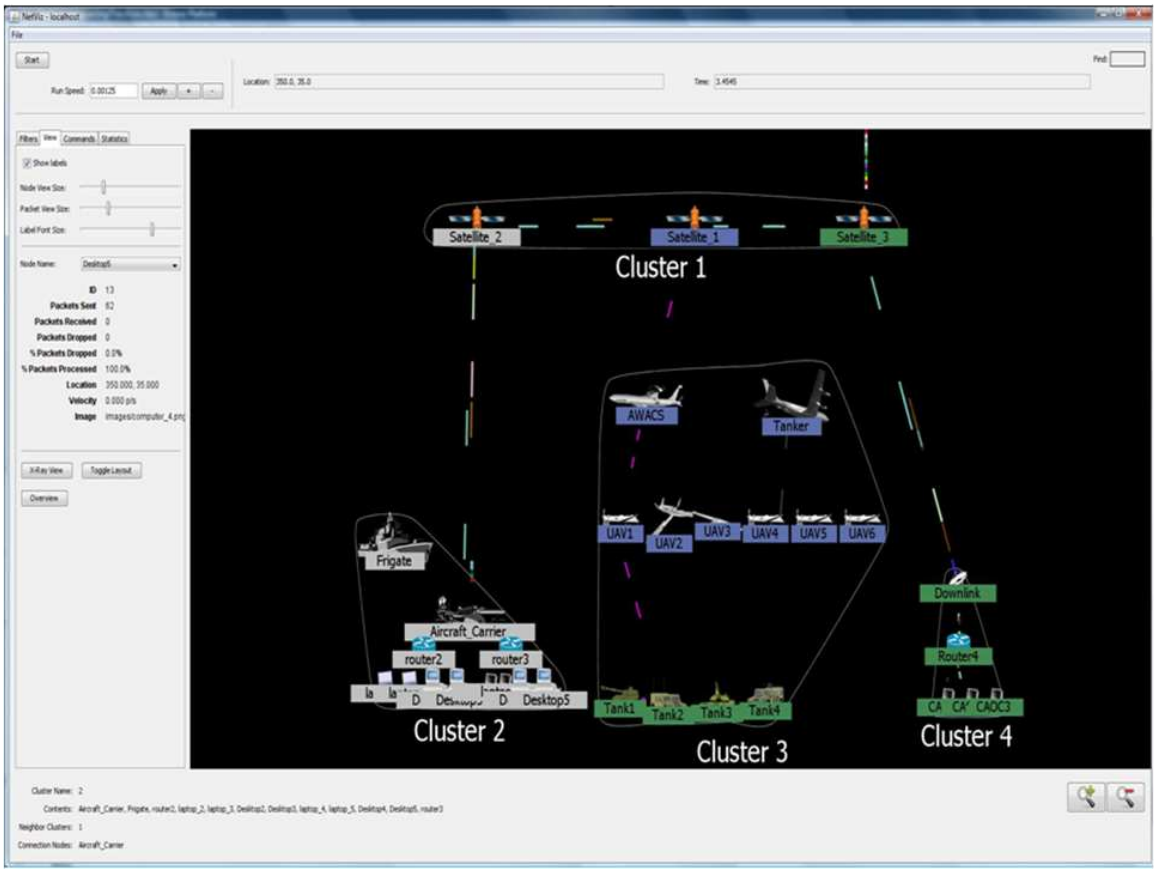


Figure 4.6: The "Show labels" check box on the view tab is deactivated to remove all the labels from the screen.

is not color-coded. However, colors of clusters and force labels are implemented to the new NetViz display.

As shown in Figure 4.2, the color customization panel provides a color scheme to specify the cluster colors on the display. For the colors of force labels, force names and keywords are added into the some of NS-2 files discussed in Section 3.4. Each force is given its own color such as green for Army, blue for Air Force, gray for Navy. These colors are used in the background of the labels texts in rectangular shapes.

Additionally, "X-Ray" mode button can be used to turn the background of the display black. Thus, the label background colors look highlighted to indicate the forces on the display. Figure 4.7 shows NetViz in "X-Ray" mode visualizing the nodes included in four clusters. Emphasizing the force colors, this mode increases the visual perception of human.



Run Speed: 0.00125

Location: 350.0, 35.0

Time: 3.4545

☑ Show labels

Node View Size:

Packet View Size:

Label Font Size:

Node Name: Desktop5

ID 13
Packets Sent 62
Packets Received 0
Packets Dropped 0
% Packets Dropped 0.0%
% Packets Processed 100.0%
Location 350.000, 35.000
Velocity 0.000 p/s
Image images/computer_4.png

Cluster Name: 2
Contents: Aircraft_Carrier, Frigate, router2, laptop_2, laptop_3, Desktop2, Desktop3, laptop_4, laptop_5, Desktop4, Desktop5, router3
Neighbor Clusters: 1
Connection Nodes: Aircraft_Carrier

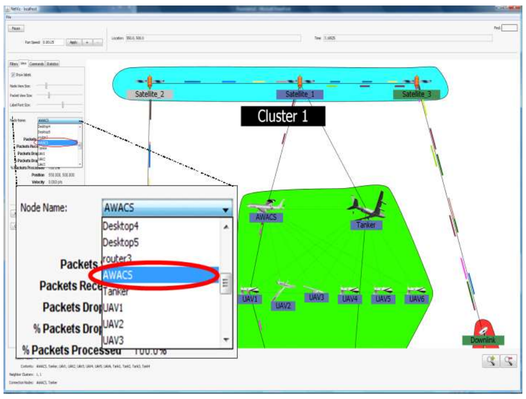
Figure 4.7: NetViz in "X-Ray" mode visualizing the JSE scenario including four clusters. "X-Ray" mode button is used to turn the background of the display black. Thus, the label background colors look highlighted to indicate the colors of forces on the display.

When the zooming and panning are performed, users can display the default view of network objects clicking on "Overview" button. "Toggle layout" button and information regarding nodes are common features with the old version of NetViz and discussed in Section 4.3.

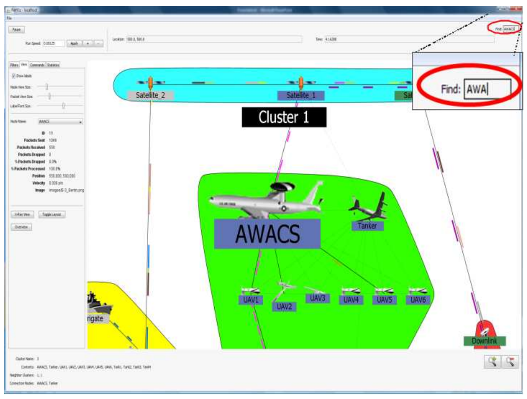
4.1.5 Selecting and Finding Nodes. In the prototype design, a number of techniques were determined to ease the complexity of the network. The "find" field was an exploration tool which is a common feature with the old version of NetViz.

But in some cases it is desirable to see all the components of the network in a list format. For this reason, a hierarchical order and pull-down menus were included in the prototype design. Instead of the hierarchical order, the new NetViz framework has the view tree on the filters tab which enables users to remove items from the display to reduce visual complexity. On the other hand, the pull-down menus are facilitated to let users to select the desired node from the list for the purposed tasks. Except filters tab each tab of NetViz has pull-down menu or menus. In the case depicted in Figure 4.8 (a), a circle indicates "AWACS" node which is selected from the list of the pull-down menu for the observation. After clicking on the desired node, the information about this node is appeared below the pull-down menu.

The find field supports keyword search for the visualized objects. As users start to type the desired node name, the name and the node will grow clearly larger if there are at least 2 matching letters. A circle in Figure 4.8 (b) shows that a user types "AWACS" node which grows larger on the display to highlight the visibility. This amplifies cognition and allows users to search and observe particular nodes in large networks.



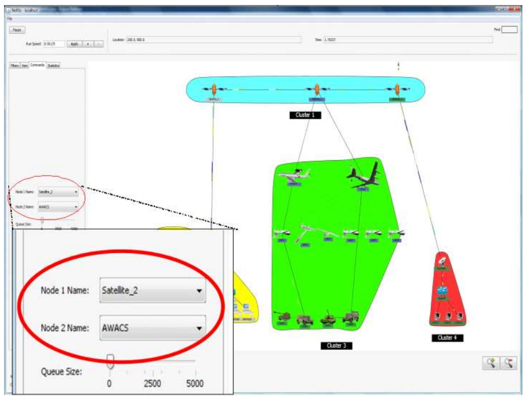
(a) The pull-down menu on the view tab. A circle indicates that "AWACS" node is selected from the list of the pull-down menu for the observation.



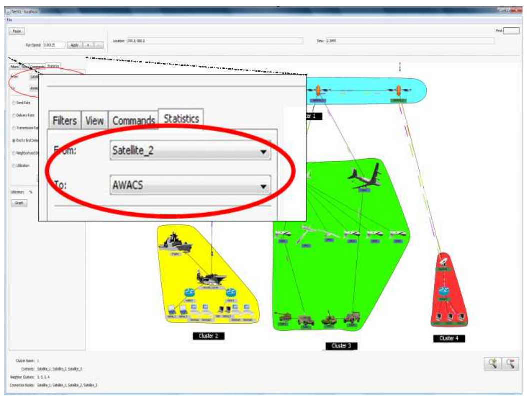
(b) The "find" field on the NetViz user interface. A circle shows that a user types "AWACS" node which grows larger on the display

Figure 4.8: Two NetViz displays on the view tab showing the ways for selecting and finding nodes.

The pull-down menus included in the command tab are shown in Figure 4.9 (a). In this case, these menus are used for accessing nodes to change specific features of them interacting with NS-2. Even though the functionality of the statistics tab is proceeding, the pull-down menus are facilitated on this tab (See Figure 4.9 (b)). When the statistics tab functionality is completed, users will be able to analyze particular node activities selecting these nodes from the pull-down menu list.



(a) Two pull-down menus on the command tab. A circle indicates two nodes which are selected from the lists of these menus to change specific features of the nodes interacting with NS-2



(b) Two pull-down menus on the statistics tab. A circle indicates two nodes which are selected from the lists of these menus, even though the functionality of this tab is done yet

Figure 4.9: The pull down menus on the command and statistic tab of the new NetViz user interface.

To accomplish the "find" field and selecting nodes the labeling function is established as mentioned in section 4.1.3. In the old NetViz framework, the task of "find" field was to search flow id numbers since the labeling function didn't exist. Usage of selecting nodes and "find" field strengthens our design giving the ability to access the desired object.

4.2 Test and Validation of NetViz-NS-2 Interaction

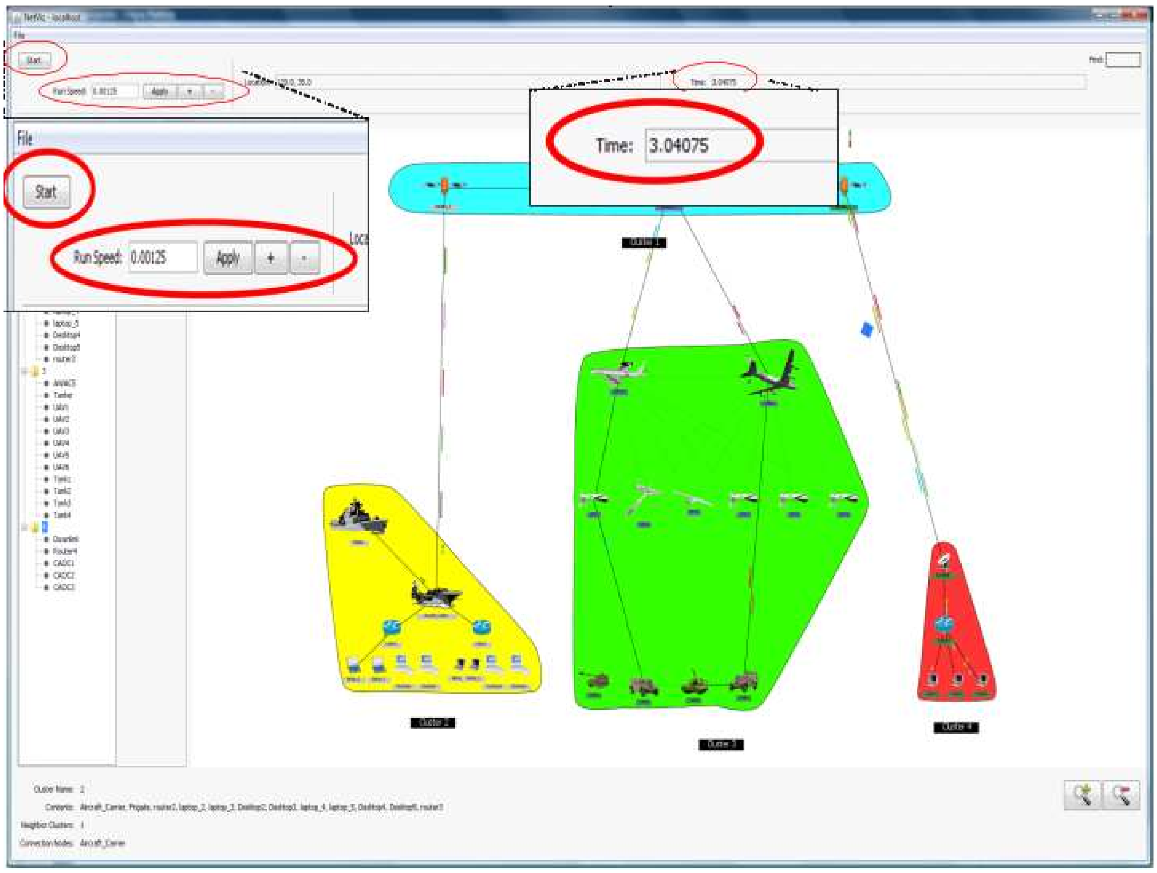
Our research used the mediator tool to build a real-time interaction between the new NetViz and modified NS-2. The interaction is established by enhancing the NetViz classes and modifying some of the NS-2 script files as described in Section 3.3.

Passing data forth and commands back between NS-2 and the new NetViz framework, the mediator tool gives users the ability to interact with NS-2, as it executes.

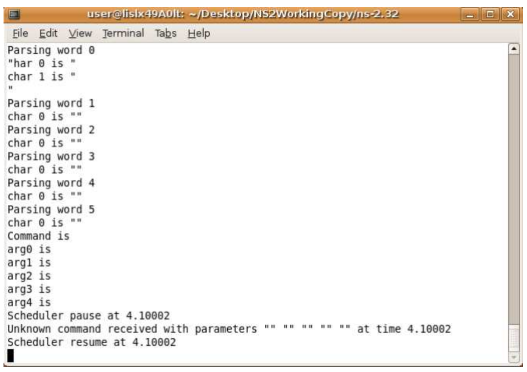
Since the new NetViz doesn't use the trace file data at start up, the data is needed to be streamed from NS-2 Tcl Script. The mediator receives this data from the modified NS-2 as it executes. When a user gives a command, NetViz passes the command through the mediator to the modified NS-2 and streams the data after 1.1 seconds which is an off-set time. Currently, "pause", "start" and "change the queue size" commands are implemented in the new NetViz. The commands are tested and validated visualizing the JSE wired scenario and a simple scenario in the following sections. The two sections illustrate the interaction capability of NetViz showing wired topologies, as NS-2 progresses.

4.2.1 Pausing and Resuming the New NetViz. Providing the real-time interaction with modified NS-2, the mediator tool brings the capability of pausing and resuming the new NetViz any time during the modified NS-2 execution. User interface of the new NetViz has controls for starting, pausing and stopping the visualization. It also provides speed control in the visualization. At the top are the control panel; the lower left side two buttons (+, -) holds the time control.The circles on visualized JSE wired scenario indicate these areas in Figure 4.10 (a).

The method of starting the new NetViz, including running the *mediator* and CCT programs, is shown in Section 3.4 and 3.5. After starting the new NetViz, it can be paused, resumed or completely stopped by user action. Clicking on the "pause" button on the user interface, users pauses the scenario flow at a certain time. At that time, modified NS-2 freezes the scenario execution as well. Figure 4.10 (a) shows the new NetViz user interface when NetViz is paused at time 3.04075 sec. As the user click on the "start" button to resume the visualization paused at this time, the command console shows the response in the Figure 4.10 (b). It shows 4.1002 sec which is the time that modified NS-2 pauses the simulation. The offset delay ensures NS-2 execution stays ahead of the visualization. For example 1.1 sec is the time

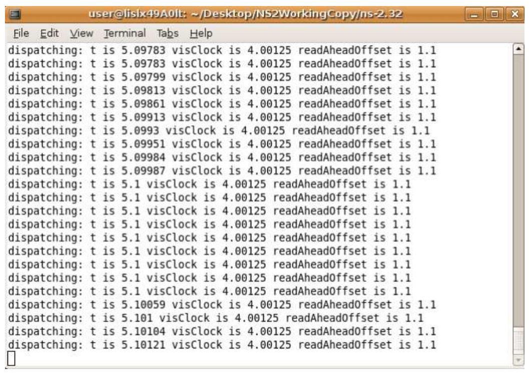

File
Start
Run Speed: 0.00125
Apply
+
-
Time: 3.04075
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster Name: 2
Contents: Aircraft_Carrier, Frigate, router2, laptop_2, laptop_3, Desktop2, Desktop3, laptop_4, laptop_5, Desktop4, Desktop5, router3
Neighbor Clusters: 1
Connection Nodes: Aircraft_Carrier

(a) The screenshot of the new NetViz user interface paused at time 3.04075


user@lislx49A0lt: ~/Desktop/NS2WorkingCopy/ns-2.32
File Edit View Terminal Tabs Help

```
Parsing word 0
"har 0 is "
char 1 is "
"
Parsing word 1
char 0 is ""
Parsing word 2
char 0 is ""
Parsing word 3
char 0 is ""
Parsing word 4
char 0 is ""
Parsing word 5
char 0 is ""
Command is
arg0 is
arg1 is
arg2 is
arg3 is
arg4 is
Scheduler pause at 4.10002
Unknown command received with parameters "" "" "" "" "" at time 4.10002
Scheduler resume at 4.10002
```

(b) Command console showing the NS-2 response when clicked on start button at time 3.04075. NS- 2 pauses the execution at 4.1002


user@lislx49A0lt: ~/Desktop/NS2WorkingCopy/ns-2.32
File Edit View Terminal Tabs Help

```
dispatching: t is 5.09783 visClock is 4.00125 readAheadOffset is 1.1
dispatching: t is 5.09783 visClock is 4.00125 readAheadOffset is 1.1
dispatching: t is 5.09799 visClock is 4.00125 readAheadOffset is 1.1
dispatching: t is 5.09813 visClock is 4.00125 readAheadOffset is 1.1
dispatching: t is 5.09861 visClock is 4.00125 readAheadOffset is 1.1
dispatching: t is 5.09913 visClock is 4.00125 readAheadOffset is 1.1
dispatching: t is 5.0993 visClock is 4.00125 readAheadOffset is 1.1
dispatching: t is 5.09951 visClock is 4.00125 readAheadOffset is 1.1
dispatching: t is 5.09984 visClock is 4.00125 readAheadOffset is 1.1
dispatching: t is 5.09987 visClock is 4.00125 readAheadOffset is 1.1
dispatching: t is 5.1 visClock is 4.00125 readAheadOffset is 1.1
dispatching: t is 5.1 visClock is 4.00125 readAheadOffset is 1.1
dispatching: t is 5.1 visClock is 4.00125 readAheadOffset is 1.1
dispatching: t is 5.1 visClock is 4.00125 readAheadOffset is 1.1
dispatching: t is 5.1 visClock is 4.00125 readAheadOffset is 1.1
dispatching: t is 5.1 visClock is 4.00125 readAheadOffset is 1.1
dispatching: t is 5.1 visClock is 4.00125 readAheadOffset is 1.1
dispatching: t is 5.1 visClock is 4.00125 readAheadOffset is 1.1
dispatching: t is 5.1 visClock is 4.00125 readAheadOffset is 1.1
dispatching: t is 5.10059 visClock is 4.00125 readAheadOffset is 1.1
dispatching: t is 5.101 visClock is 4.00125 readAheadOffset is 1.1
dispatching: t is 5.10104 visClock is 4.00125 readAheadOffset is 1.1
dispatching: t is 5.10121 visClock is 4.00125 readAheadOffset is 1.1
```

(c) NS-2 continues to normal execution.

Figure 4.10: Snapshots of the new NetViz user interface and command console of NS-2 when pausing and resuming the visualization.

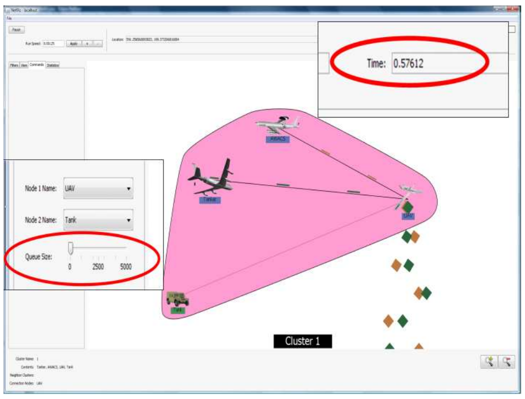
difference that is set in the NetViz configuration file as offset time. Then modified

NS-2 continues to normal execution as shown in Figure 4.10 (c). Users can stop the visualization simply by closing the application window.

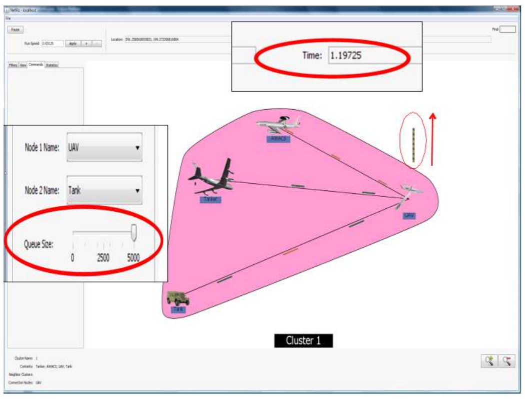
4.2.2 Changing the Queue Size. The command tab of the prototype design is planned to have multiple commands in Chapter III. Currently, the only command on the command tab that can be received and processed by the modified NS-2 is "change queue size". Coupling the additional panel, CCT, with the new NetViz, [12] run a set of additional commands successfully. These commands are discussed for the future work in Chapter V.

The command tab functionality for "change queue size" command is tested visualizing JSE scenario (Shown in Figure 4.11 and 4.12). These figures illustrate a possible real-world scenario for monitoring a military network including AWACS, Tanker, UAV and Tank nodes. Figure 4.11 (a) shows the initial network with two links up and queue size of UAV is zero. In this case, UAV communicates with AWACS and Tanker via receiving packets. Since the queue size is zero, there is no communication between UAV and Tanker and thus, all the packets drop. As shown in Figure 4.11(b), after selecting UAV and Tank nodes from pull-down menus lists, the cursor of "queue slider" is dragged to the right to increase queue size from zero to 5000. This command is given at time 0.17612. Figure 4.11 (c) shows the modified NS-2 response to this command. Then NS-2 continues to the execution (Figure 4.11 (d)). The new NetViz streams data with a delay based on the user-defined offset. Then the display shows the result of the given command (see Figure 4.11(b)). As the queue buffers, the UAV's queue begins to spike from the network traffic and starts sending packets to the Tank. Thus, there is no dropped packet.

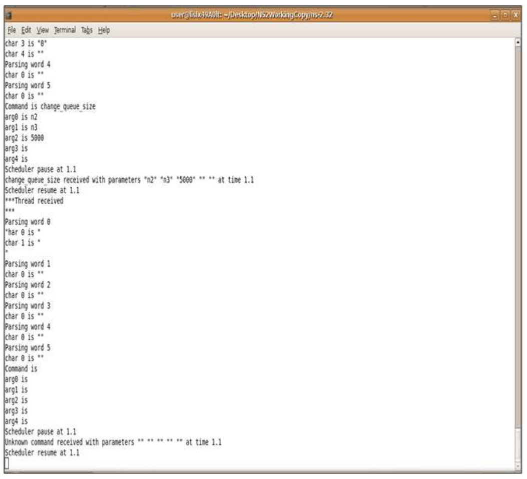
The cursor of "queue slider" is dragged to the left to decrease queue size from 5000 to zero at time 1.16525 as shown in Figure 4.12 (a). Figure 4.12 (c) shows that the command is received and processed by the modified NS-2. Then NS-2 continues to the execution (Figure 4.12 (d)). As shown in Figure 4.12 (b), UAV continues to communicate with Tank. As the queue flushes, the packets begin to drop again. When



(a) The command tab of the new NetViz user interface showing the queue size of UAV node is zero in the circle. UAV communicates with AWACS and Tanker via receiving packets. Since the queue size is zero, there is no communication between UAV and Tanker and thus, all the packets drop.

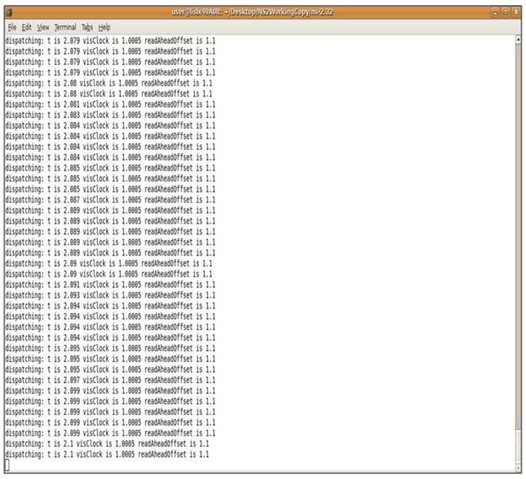


(b) The command tab of the new NetViz user interface showing the queue size is changed from zero to 5000 at time 0.17612. As the queue on, UAV begins to spike from the network traffic and starts sending packets to the Tank. Thus, there is no dropped packet at time 1.19725.


user@lislx49A0lt: ~/Desktop/NS2WorkingCopy/ns-2.32
File Edit View Terminal Tabs Help

```
char 3 is "0"
char 4 is ""
Parsing word 4
char 0 is ""
Parsing word 5
char 0 is ""
Command is change_queue_size
arg0 is n2
arg1 is n3
arg2 is 5000
arg3 is
arg4 is
Scheduler pause at 1.1
change_queue_size received with parameters "n2" "n3" "5000" "" "" at time 1.1
Scheduler resume at 1.1
***Thread received
***
Parsing word 0
"har 0 is "
char 1 is "
"
Parsing word 1
char 0 is ""
Parsing word 2
char 0 is ""
Parsing word 3
char 0 is ""
Parsing word 4
char 0 is ""
Parsing word 5
char 0 is ""
Command is
arg0 is
arg1 is
arg2 is
arg3 is
arg4 is
Scheduler pause at 1.1
Unknown command received with parameters "" "" "" "" "" at time 1.1
Scheduler resume at 1.1
```

(c) Command console showing the NS-2 response when the queue size is changed from zero to 5000.


user@lislx49A0lt: ~/Desktop/NS2WorkingCopy/ns-2.32
File Edit View Terminal Tabs Help

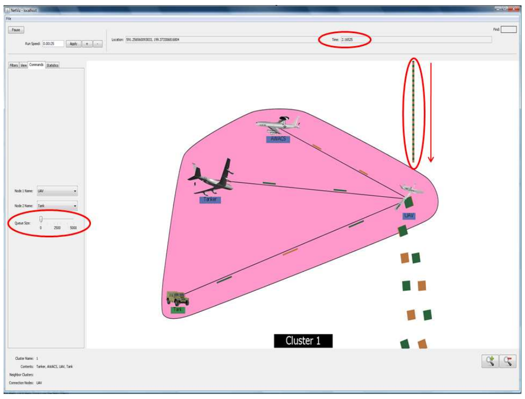
```
dispatching: t is 2.079 visClock is 1.0005 readAheadOffset is 1.1
dispatching: t is 2.079 visClock is 1.0005 readAheadOffset is 1.1
dispatching: t is 2.079 visClock is 1.0005 readAheadOffset is 1.1
dispatching: t is 2.079 visClock is 1.0005 readAheadOffset is 1.1
dispatching: t is 2.08 visClock is 1.0005 readAheadOffset is 1.1
dispatching: t is 2.08 visClock is 1.0005 readAheadOffset is 1.1
dispatching: t is 2.081 visClock is 1.0005 readAheadOffset is 1.1
dispatching: t is 2.083 visClock is 1.0005 readAheadOffset is 1.1
dispatching: t is 2.084 visClock is 1.0005 readAheadOffset is 1.1
dispatching: t is 2.084 visClock is 1.0005 readAheadOffset is 1.1
dispatching: t is 2.084 visClock is 1.0005 readAheadOffset is 1.1
dispatching: t is 2.084 visClock is 1.0005 readAheadOffset is 1.1
dispatching: t is 2.085 visClock is 1.0005 readAheadOffset is 1.1
dispatching: t is 2.085 visClock is 1.0005 readAheadOffset is 1.1
dispatching: t is 2.085 visClock is 1.0005 readAheadOffset is 1.1
dispatching: t is 2.087 visClock is 1.0005 readAheadOffset is 1.1
dispatching: t is 2.089 visClock is 1.0005 readAheadOffset is 1.1
dispatching: t is 2.089 visClock is 1.0005 readAheadOffset is 1.1
dispatching: t is 2.089 visClock is 1.0005 readAheadOffset is 1.1
dispatching: t is 2.089 visClock is 1.0005 readAheadOffset is 1.1
dispatching: t is 2.089 visClock is 1.0005 readAheadOffset is 1.1
dispatching: t is 2.09 visClock is 1.0005 readAheadOffset is 1.1
dispatching: t is 2.09 visClock is 1.0005 readAheadOffset is 1.1
dispatching: t is 2.091 visClock is 1.0005 readAheadOffset is 1.1
dispatching: t is 2.093 visClock is 1.0005 readAheadOffset is 1.1
dispatching: t is 2.094 visClock is 1.0005 readAheadOffset is 1.1
dispatching: t is 2.094 visClock is 1.0005 readAheadOffset is 1.1
dispatching: t is 2.094 visClock is 1.0005 readAheadOffset is 1.1
dispatching: t is 2.094 visClock is 1.0005 readAheadOffset is 1.1
dispatching: t is 2.095 visClock is 1.0005 readAheadOffset is 1.1
dispatching: t is 2.095 visClock is 1.0005 readAheadOffset is 1.1
dispatching: t is 2.095 visClock is 1.0005 readAheadOffset is 1.1
dispatching: t is 2.097 visClock is 1.0005 readAheadOffset is 1.1
dispatching: t is 2.099 visClock is 1.0005 readAheadOffset is 1.1
dispatching: t is 2.099 visClock is 1.0005 readAheadOffset is 1.1
dispatching: t is 2.099 visClock is 1.0005 readAheadOffset is 1.1
dispatching: t is 2.099 visClock is 1.0005 readAheadOffset is 1.1
dispatching: t is 2.099 visClock is 1.0005 readAheadOffset is 1.1
dispatching: t is 2.1 visClock is 1.0005 readAheadOffset is 1.1
dispatching: t is 2.1 visClock is 1.0005 readAheadOffset is 1.1
```

(d) NS-2 continues to normal execution.

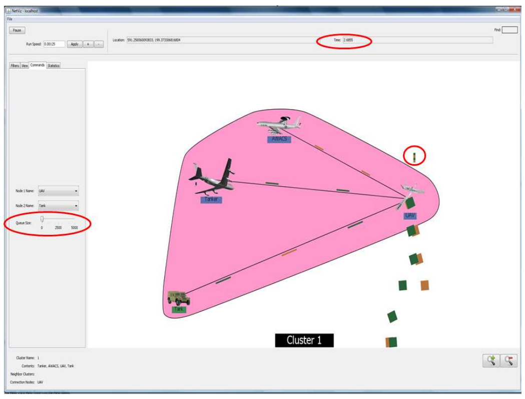
Figure 4.11: Snapshots of the command tab of the new NetViz user interface and command console of NS-2 when increasing queue size of the nodes. The displays illustrate a real-world scenario for monitoring a military network including AWACS, Tanker, UAV and Tank nodes.

the spike is gone, the queue size will be zero. Thus, the communication between UAV and Tanker will go down.

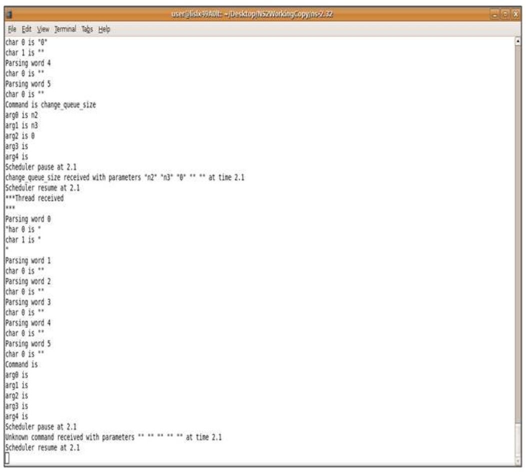
The view tab of the new NetViz user interface is shown in Figure 4.13. The display at time 3.0020 shows a similar view of the scenario to the initial case in Figure



(a) The command tab of the new NetViz user interface showing the queue size is changed from 5000 to zero at time 1.16525. While UAV continues to communicate with Tank, as the queue down, the packets begin to drop.



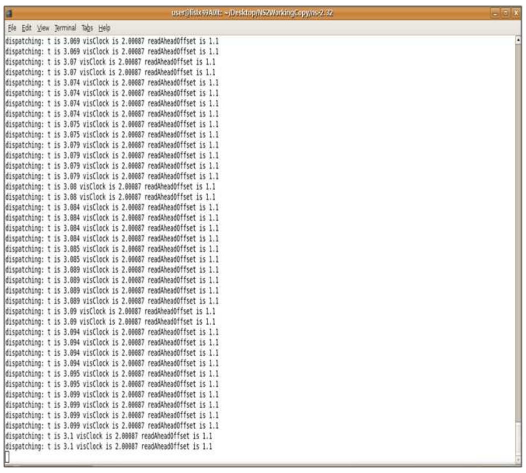
(b) The command tab of the new NetViz user interface showing the queue size is zero again. The queue length is small. When the spike is gone, the queue size will be zero and the communication between UAV and Tanker finish again.



```
user@lislx49A0lt: ~/Desktop/NS2WorkingCopy/ns-2.32
File  Edit  View  Terminal  Tabs  Help
char 0 is "0"
char 1 is ""
Parsing word 4
char 0 is ""
Parsing word 5
char 0 is ""
Command is change_queue_size
arg0 is n2
arg1 is n3
arg2 is 0
arg3 is
arg4 is
Scheduler pause at 2.1
change_queue_size received with parameters "n2" "n3" "0" "" "" at time 2.1
Scheduler resume at 2.1
***Thread received
***
Parsing word 0
"har 0 is "
char 1 is "
"

Parsing word 1
char 0 is ""
Parsing word 2
char 0 is ""
Parsing word 3
char 0 is ""
Parsing word 4
char 0 is ""
Parsing word 5
char 0 is ""
Command is
arg0 is
arg1 is
arg2 is
arg3 is
arg4 is
Scheduler pause at 2.1
Unknown command received with parameters "" "" "" "" "" at time 2.1
Scheduler resume at 2.1
```

(c) Command console showing the NS-2 response when the queue size is changed from 5000 to zero.



```
user@lislx49A0lt: ~/Desktop/NS2WorkingCopy/ns-2.32
File  Edit  View  Terminal  Tabs  Help
dispatching: t is 3.069 visClock is 2.00087 readAheadOffset is 1.1
dispatching: t is 3.069 visClock is 2.00087 readAheadOffset is 1.1
dispatching: t is 3.07 visClock is 2.00087 readAheadOffset is 1.1
dispatching: t is 3.07 visClock is 2.00087 readAheadOffset is 1.1
dispatching: t is 3.074 visClock is 2.00087 readAheadOffset is 1.1
dispatching: t is 3.074 visClock is 2.00087 readAheadOffset is 1.1
dispatching: t is 3.074 visClock is 2.00087 readAheadOffset is 1.1
dispatching: t is 3.074 visClock is 2.00087 readAheadOffset is 1.1
dispatching: t is 3.075 visClock is 2.00087 readAheadOffset is 1.1
dispatching: t is 3.075 visClock is 2.00087 readAheadOffset is 1.1
dispatching: t is 3.079 visClock is 2.00087 readAheadOffset is 1.1
dispatching: t is 3.079 visClock is 2.00087 readAheadOffset is 1.1
dispatching: t is 3.079 visClock is 2.00087 readAheadOffset is 1.1
dispatching: t is 3.079 visClock is 2.00087 readAheadOffset is 1.1
dispatching: t is 3.08 visClock is 2.00087 readAheadOffset is 1.1
dispatching: t is 3.08 visClock is 2.00087 readAheadOffset is 1.1
dispatching: t is 3.084 visClock is 2.00087 readAheadOffset is 1.1
dispatching: t is 3.084 visClock is 2.00087 readAheadOffset is 1.1
dispatching: t is 3.084 visClock is 2.00087 readAheadOffset is 1.1
dispatching: t is 3.084 visClock is 2.00087 readAheadOffset is 1.1
dispatching: t is 3.085 visClock is 2.00087 readAheadOffset is 1.1
dispatching: t is 3.085 visClock is 2.00087 readAheadOffset is 1.1
dispatching: t is 3.089 visClock is 2.00087 readAheadOffset is 1.1
dispatching: t is 3.089 visClock is 2.00087 readAheadOffset is 1.1
dispatching: t is 3.089 visClock is 2.00087 readAheadOffset is 1.1
dispatching: t is 3.089 visClock is 2.00087 readAheadOffset is 1.1
dispatching: t is 3.09 visClock is 2.00087 readAheadOffset is 1.1
dispatching: t is 3.09 visClock is 2.00087 readAheadOffset is 1.1
dispatching: t is 3.094 visClock is 2.00087 readAheadOffset is 1.1
dispatching: t is 3.094 visClock is 2.00087 readAheadOffset is 1.1
dispatching: t is 3.094 visClock is 2.00087 readAheadOffset is 1.1
dispatching: t is 3.094 visClock is 2.00087 readAheadOffset is 1.1
dispatching: t is 3.095 visClock is 2.00087 readAheadOffset is 1.1
dispatching: t is 3.095 visClock is 2.00087 readAheadOffset is 1.1
dispatching: t is 3.099 visClock is 2.00087 readAheadOffset is 1.1
dispatching: t is 3.099 visClock is 2.00087 readAheadOffset is 1.1
dispatching: t is 3.099 visClock is 2.00087 readAheadOffset is 1.1
dispatching: t is 3.099 visClock is 2.00087 readAheadOffset is 1.1
dispatching: t is 3.1 visClock is 2.00087 readAheadOffset is 1.1
dispatching: t is 3.1 visClock is 2.00087 readAheadOffset is 1.1
```

(d) NS-2 continues to normal execution.

Figure 4.12: Snapshots of the command tab of the new NetViz user interface and command console of NS-2 when decreasing queue size of the nodes. The displays illustrate a real-world scenario for monitoring a military network including AWACS, Tanker, UAV and Tank nodes.

4.11. But the information regarding Tank node on this tab indicates that the number of the received packet is 398 which was 0 in the beginning. Without the interaction with NS-2, there is no way to increase the number of packets of Tank node in a static scenario. This verifies the real-time interaction capability of the new NetViz with the modified NS-2 using *mediator* tool.

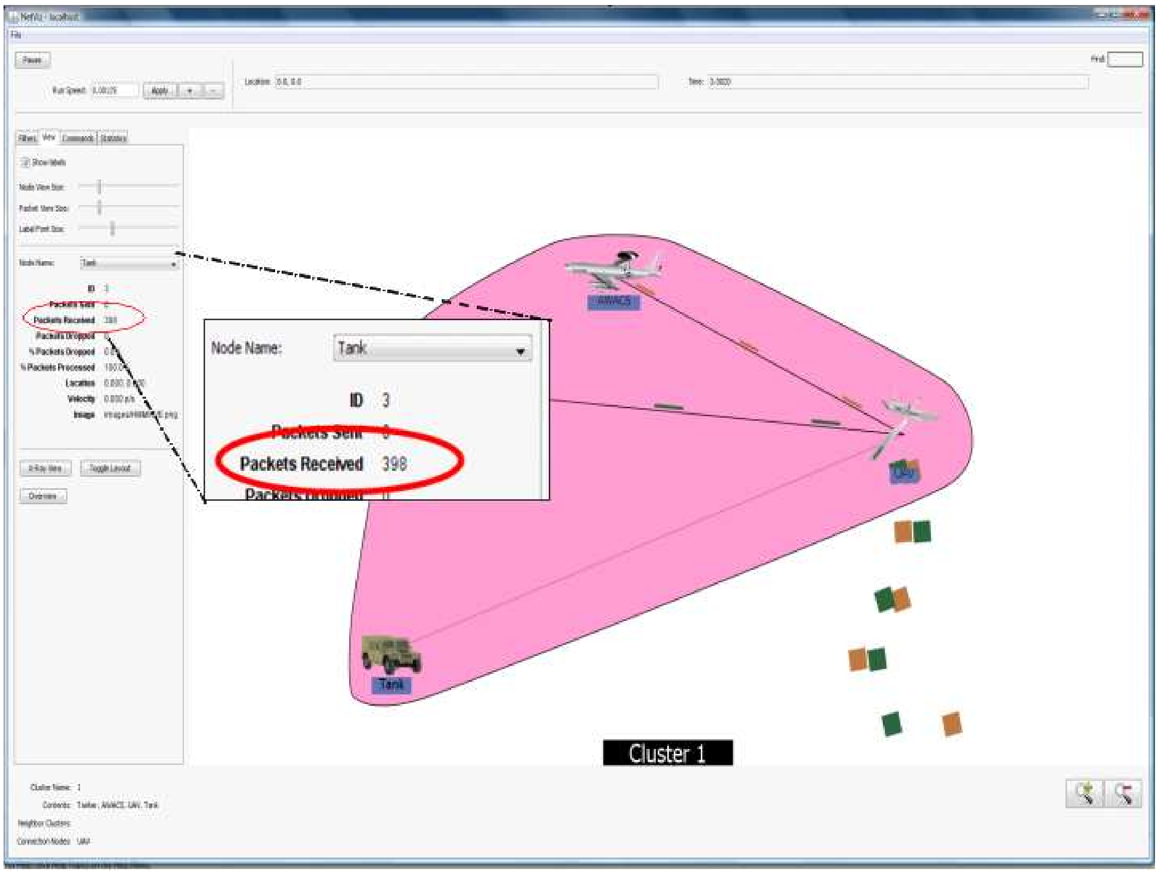


Figure 4.13: Snapshots of the command tab of the new NetViz user interface displaying a real-world scenario at time 3.0020 for monitoring a military network including AWACS, Tanker, UAV and Tank nodes. Information regarding Tank node on this tab indicates that the number of the received packet is 398 which was 0 in the beginning.

4.3 Comparison to NAM and The Old NetViz Framework

NAM and the old version of NetViz framework operate similarly; process trace files generated during NS-2 execution, translate the data, and display the information to a monitor. Our framework has two big differences from NAM and the old version of NetViz. First, the new NetViz framework provides user interaction with NS-2 using the *mediator* tool. The second difference is the way the data presentation of large network scenarios using visualization functions such as clustering, labeling and color coding. These differences were tested and validated presenting the results in previous sections. Comparison of the capabilities of our framework to NAM and the old NetViz framework are discussed in the following sections.

4.3.1 Comparison to NAM. After the NS-2 execution, NAM parses the trace file (file.nam) built during the execution as discussed in Section 3.2. It can also be executed directly from a Tcl script. The only user interaction with NAM is limited to setting up network scenarios. But it does not have the capability to let the user interact the system while a scenario is running. The new NetViz framework presented in this research is coupled with the *mediator* tool and has the capability of rendering the simulation during the execution. The modified NS-2 executes the Tcl script. The new NetViz streams the data received from the modified NS-2. As the data is streamed and the scenario is changed, the received data is added and modified dynamically to the new NetViz. While the new NetViz has all the interaction capabilities presented in Section 4.2, NAM has the ability to introduce a static information obtained from the trace file including the simulation results.

NAM has some user interface capabilities such as fast-forward/rewind, pause, jump slider as discussed in Section 2.1.2. These capabilities assist the user during analysis of network events during the visualization. To alter specific visualization characteristics and simplify the visual presentation NAM does not include any visualization functions. On the other hand, the new NetViz has visualization functions such as clustering, labeling, color coding which assist in accessing network objects and events, supporting four tabs consisting of buttons, menus, and sliders.

Figure 4.14 and 4.15 show the user interfaces of the NAM and the new NetViz. Both visualization softwares visualize JSE scenario and show the network visualization at time 9.06 seconds. As shown in Figure 4.14, NAM is lack of capabilities discussed in Section 4.1. The display of NAM does not include visualization functions, filtering, labeling, color coding. It does not have comprehensive interactive techniques such as menus, sliders either. Even though there are names refer to the nodes on the display,

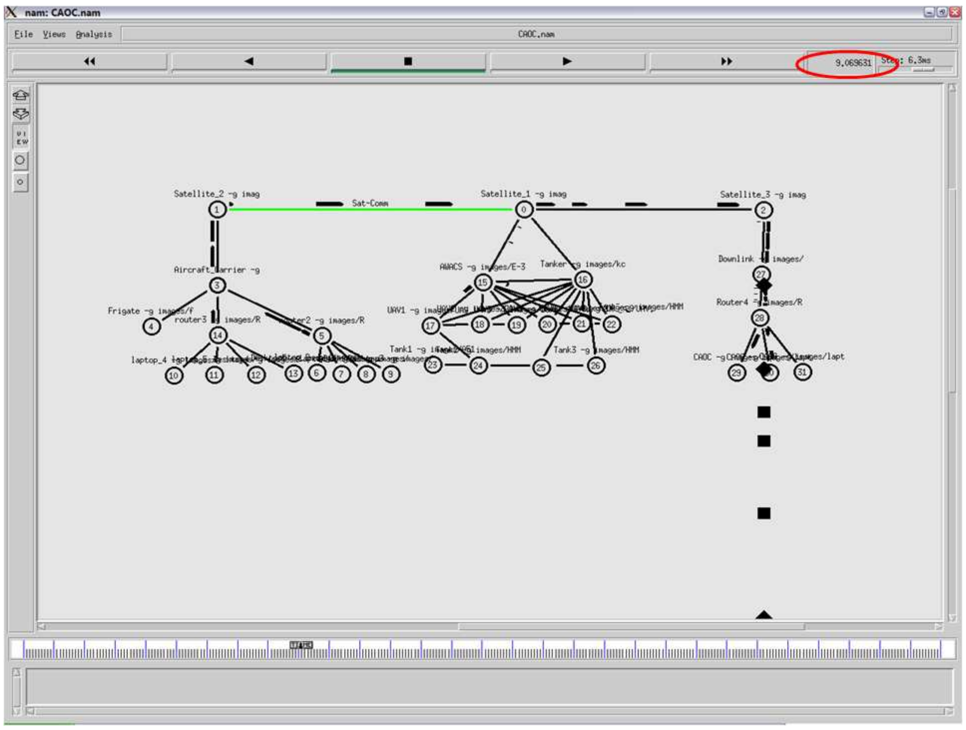
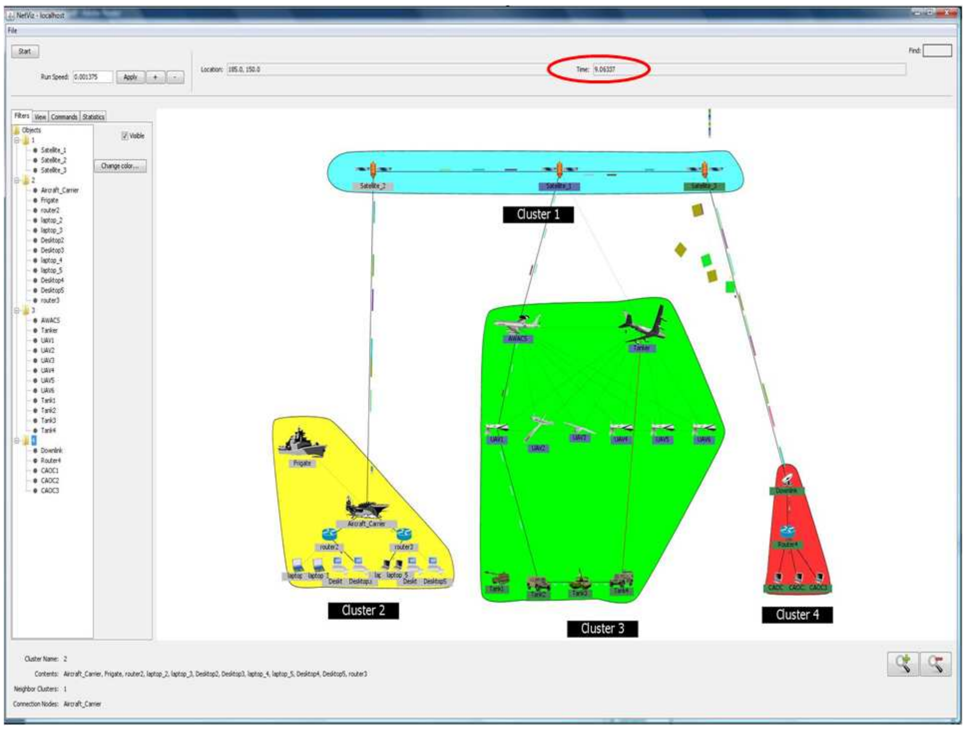


Figure 4.14: The screenshot of NAM visualizing JSE wired scenario paused at time 9.06.

there is a disorder with these names including unneeded letters. The reason for this, NAM parses these names from image names, which does not serve to the labeling function directly. NAM does not use these images to indicate nodes either. Figure 4.15 shows the filters tab of the new NetViz framework including all the nodes in cluster structures.

Both NAM and the new NetViz provide packet-level animation. NAM can graphically present information such as throughput and number of packets dropped at each link and the NetViz provides node statistics such as packet receives, drops, and receive/drop percentages, when desired node is selected from the pull-down menu of the view tab. Since our framework similar to the old version of NetViz by the means of this feature, [5] can provide useful information about it.

4.3.2 Comparison to Old Version of NetViz. The first difference of the old NetViz framework from our framework is being lack of capability of interacting


Run Speed: 0.001375

Location: 185.0, 150.0

Time: 9.06337

Cluster Name: 2

Contents: Aircraft_Carrier, Frigate, router2, laptop_2, laptop_3, Desktop2, Desktop3, laptop_4, laptop_5, Desktop4, Desktop5, router3

Neighbor Clusters: 1

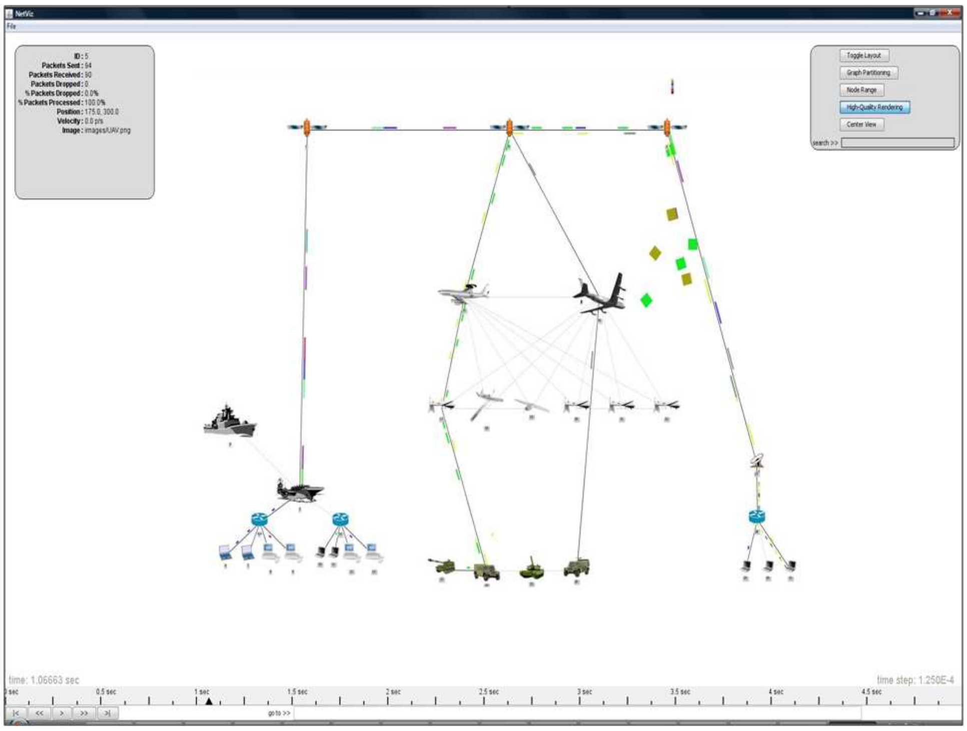
Connection Nodes: Aircraft_Carrier

Figure 4.15: The screenshot of the new NetViz framework visualizing JSE wired scenario paused at time 9.06. It shows the filters tab on the left hand side and cluster structures on the right hand side.

with NS-2. The old NetViz framework uses NS-2 trace files (out.nam) obtained from a completed NS-2 execution to examine network visualization performance. Since primary classes of the old NetViz framework are kept same, our framework also can use these trace files. Because we integrated NetViz with the *mediator* and modified NS-2 code and established a complete simulation execution in real time, there is no need to use out.nam file. As modified NS-2 executes the Tcl script, our framework streams the data received from NS-2. After the data is streamed and the scenario is changed, the received data is added and modified dynamically to our framework. As described in Section 3.4, to ensure NS-2 stays ahead of NetViz, we set an offset time which is 1.1 seconds in the configuration file. The interaction capability of our framework is tested and validated in Section 4.2.

All the distinct features of our framework described in Section 4.1 also indicate differences in user interface between our framework and the old NetViz. Figure 4.16

and 4.17 show our framework and the old NetViz animating the JSE scenario with screenshots taken from both visualizations at time 1.06 seconds. The old NetViz lacks the capability of labeling, filtering, clustering, color coding and selecting nodes (See Figure 4.16). As mentioned in Section 3.3, a new format of Tcl file was required for the modified NS-2 code to provide the visualization functions in our framework. These functions are accomplished by the usage of interactive techniques presented in Section 4.1 (See Figure 4.17). Thus, these functions enable users to access network objects and events, supporting four tabs consisting of buttons, menus, and sliders providing access to variety of network objects for different tasks.

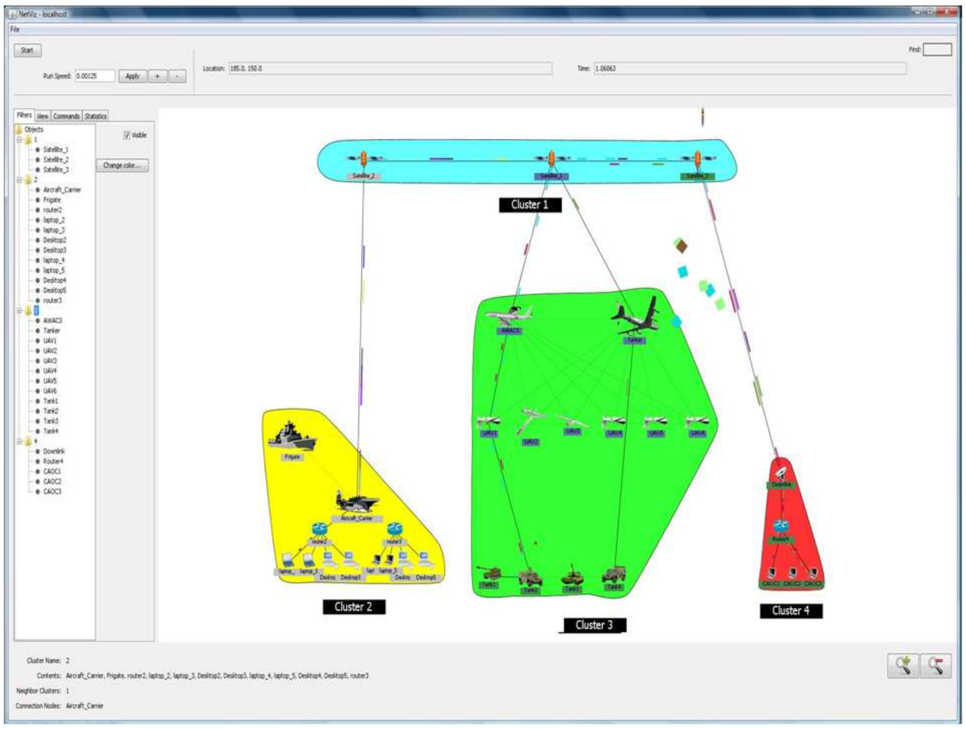

ID: 5
Packets Sent: 94
Packets Received: 90
Packets Dropped: 0
% Packets Dropped: 0.0%
% Packets Processed: 100.0%
Position: 175.0, 300.0
Velocity: 0.0 p/s
Image: images/UAV.png

time: 1.06663 sec

time step: 1.250E-4

Figure 4.16: The old version of NetViz screenshot visualizing JSE scenario and showing wired packet animation and queuing. It is lack of capability of labeling, filtering, clustering color coding.

Both the new and old NetViz provide packet-level animation. Thus, users can access node statistics such as packet receives, drops, and receive/drop percentages using different interactive techniques in both visualization suites. While the old NetViz represents this information on the display, the new NetViz framework presents this in-



Run Speed: 0.00125

Location: 185.0, 150.0

Time: 1.06063

Cluster Name: 2

Contents: Aircraft_Carrier, Frigate, router2, laptop_2, laptop_3, Desktop2, Desktop3, laptop_4, laptop_5, Desktop4, Desktop5, router3

Neighbor Clusters: 1

Connection Nodes: Aircraft_Carrier

Figure 4.17: The new version of NetViz framework's visualization of wired packet traffic visualizing JSE scenario.

formation on the view tab after selecting the desired node from the pull-down menu. The ability to alter network events at run time takes our framework to a different level. For instance, in Section 4.2, the number of received packets of Tank node was increased changing the queue size parameter, as NS-2 executes.

Overall, the old NetViz is capable of handling simple network events and has shortfalls in dynamic user interactions. On the other hand the new NetViz has the ability to handle the inherent complexity of large networks, allowing the user to interact with the current display of the framework and control the network through the visualization. Therefore, the new NetViz framework including visualization functions advances network visualization standards interacting with NS-2.

4.4 Test of the Execution Scenarios

This section tests the executions scenarios demonstrated in the Section 3.5. First execution scenario includes two NetViz clients. Then the number of NetViz clients is increased to six and the second execution scenario is obtained. The NetViz clients run on different operating system are used in this application. Establishing the second scenario it's expected that multiple NetViz clients can be linked to NS-2 using *mediator* even if they work on different operating systems. Accomplishing the interaction between the elements of these scenarios successfully provides users of NetViz clients with the capability of displaying same simulation and customizing the display differently in the direction of their needs.

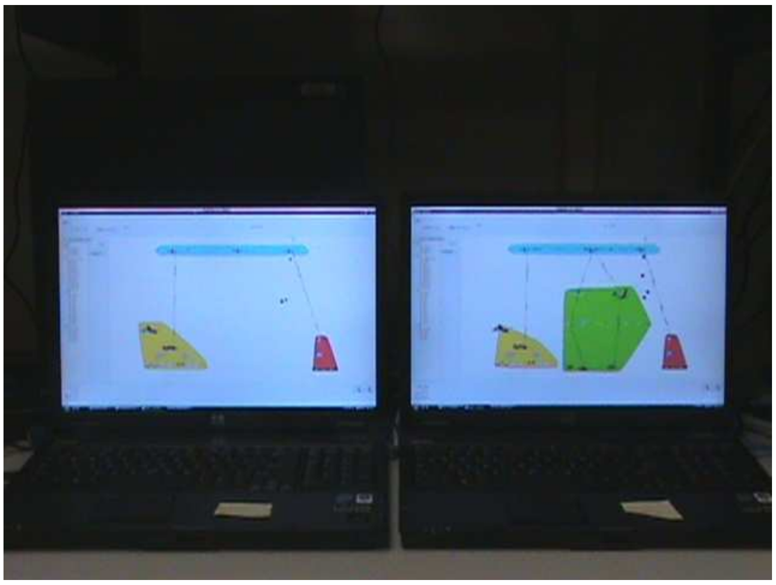


Figure 4.18: Actual picture of the first execution scenario which is composed of NS-2 running on linux operating system, the *mediator*, and two NetViz clients running windows operating system. Both clients display the filters tab. While the client on the right side has all the clusters on the screen, the cluster 3 has been removed from the display on the left side.

4.4.1 First Execution Scenario. In Figure 4.18, 4.19, a simulation execution scenario is built linking two NetViz clients to a modified NS-2 using the *mediator* tool. NetViz clients and the *mediator* tool run on windows machines and modified NS-2 code running on linux machine. This simulation execution scenario is run following the order of running programs defined Section 3.5.

The *mediator* tool collects commands from each connected NetViz client and passes them to NS-2. Then it receives data from NS-2 and duplicates it for each

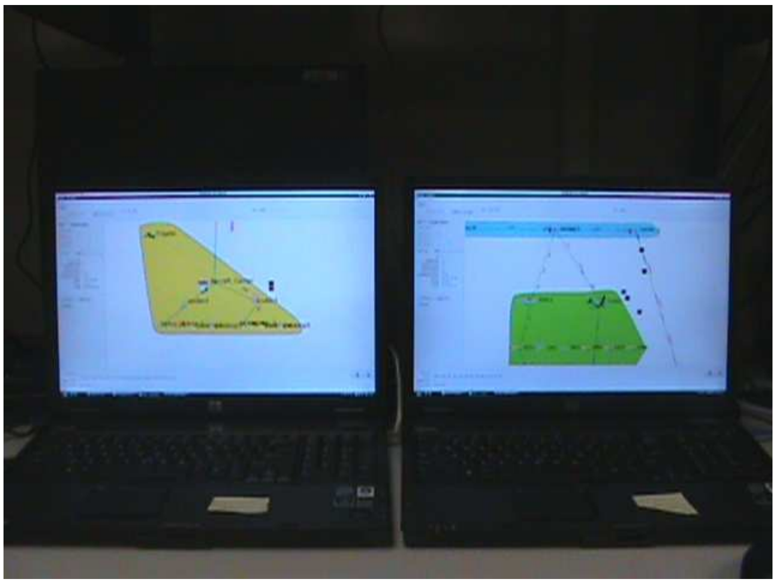


Figure 4.19: Actual picture of the first execution scenario. Both clients display the view tab. Zoom in is performed to see the cluster 2 in detail on the display of the left side. And it has larger labels and smaller images. Zooming and panning is performed to get a close view of the cluster 3. Labels and images are normal sized but packets are bigger.

NetViz client. This provides users of NetViz with the capability of displaying same simulation and customizing the display differently in the direction of needs.

Figure 4.18 shows displays of two NetViz clients. Both clients display the filters tab. The client on the left side has all the clusters on the screen. The user on the right side deactivates visible check-box for the cluster 3 and removes it from the display. In figure 4.19, both clients display the view tab. The user on the left side performs “zoom in” to see the cluster 2 in detail. And he/she also changes the view sizes of labels and images using the sliders. Thus the display shows larger labels and smaller images. On the right side, zooming and panning is performed to get a close view of the cluster 3. The user changes the view sizes of packets. Hence, labels and images are in normal size but packets are bigger.

4.4.2 Second Execution Scenario. A distributed simulation execution scenario is established adding the previous scenario four more NetViz clients. The simulation execution scenario is run following the order of running programs defined

Section 3.5. All the modifications described in Section 3.3 is done before execution starts.

Figure 4.20 shows this distributed execution scenario composed of NS-2 running on linux operating scenario, the *mediator* and five NetViz clients running on windows operating scenario and a NetViz client running on Mac operating system. This proves that multiple NetViz clients running on separate operating systems can be linked to NS-2. Additionally, as presented in the fist execution scenario each NetViz user has the capability of customizing the display changing the visualization parameters. In Figure 4.20 , each NetViz user performed a different feature of NetViz simultaneously.

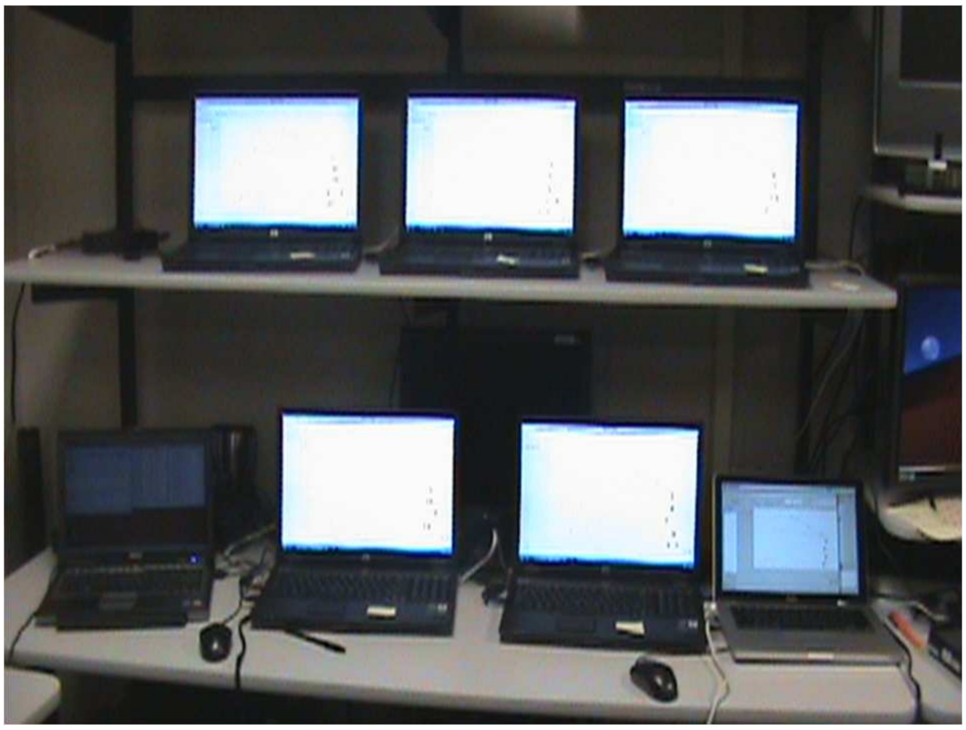


Figure 4.20: Actual picture of the second execution scenario which is composed of NS-2 running on linux operating system, the *mediator*, five NetViz clients running on windows operating system and one NetViz client running on Mac operating system.

4.5 Challenges in the Design

There was one main challenge in developing NetViz: the visualization library it uses. *Prefuse*, is good at displaying static data that is loaded at startup and then visualized. But in the case of NetViz, it does not have that data at startup. It must stream it from NS-2. To complicate matters further, data is added and modified dynamically as the trace file is streamed and the scenario changed. This made development more difficult since *prefuse* was not designed with such dynamism in mind. Certain design compromises and decisions in NetViz reflect that difficulty.

Additionally, NS-2 did not originally support some of the properties needed to display meaningful data such as cluster and force. In order to use those properties, NS-2 had to be modified to know how to pass those properties from the scenario file to the output trace file. These modifications required splitting up the node creation information from one trace line into multiple trace lines. This meant node information was not received all at once, but was instead received piece by piece. The parser had to be flexible enough to receive a node's label in one line and then its cluster a few lines later.

V. Conclusions

This research presents development and implementation of a new NetViz framework which provides visualization functions and dynamic interactions using the *prefuse* tolkit. The user interface of NetViz is effective at helping reveal cluster structures and visualization functions. The NetViz has also met performance requirements, allowing exploration of massive networks while supporting real-time interaction and animation. The interaction with NS-2 is provided by using the *mediator* tool. The following sections present contributions, research limits and future work.

5.1 Contributions

5.1.1 User Interface Design. Searching the network event in a large magnitude of network topology causes large delays leading to lower quality user interface. We investigated network interface studies, interactive techniques and the use of network visualization functions to produce a prototype design sticking to the rules of the Heuristic Evaluation Guidelines. The prototype design eased the implementation process of the new NetViz framework. The main objective of our user interface design effort in this research was to build an interface adaptable in the future to handle extremely large networks.

The prototype design was supported by four tabs ("filters", "view", "command" and "statistics") consisting of buttons, menus, and sliders providing access to variety of network objects for different tasks. Task areas were determined as controlling, optimizing, changing and analyzing the network. Because this framework builds on the robust *prefuse* visualization toolkit we used many options to enhance network visualization layouts based on the prototype design. To give users the ability to perform the tasks, we facilitated the clustering, filtering, labeling and color coding functions modifying the NS-2 files such as topo.tcl and ns-namsupp.tcl. These functions reduce the visual complexity of real world network scenarios visualized.

This research used the prototype design as a guide for the implementation process of the new NetViz design. Before constructing the new NetViz framework, this design provided a basis for the standards of interactivity and guided the development process defining visual style of the user interface. However, the final framework design has tolerable differences from the prototype design because of the limits of NS-2 and *prefuse* toolkit. For instance, while the filters tab includes a hierarchical order and filters setting table in the prototype design, the new NetViz has a check-box and tree view including nodes. The reason for this was the Java swing tables were not flexible enough to provide this capability. The command and statistics tabs are also different from the prototype design. These differences are discussed in the following sections.

5.1.2 Interaction with NS-2. To provide real-time interaction with NS-2, our research used the *mediator* tool developed by Major John Weir [12]. The *mediator* tool passes data forth and commands back between NS-2 and the new NetViz framework. The connection was established enhancing existing NetViz classes to support TCP/IP connection to be provided by the *mediator*. With this connection, instead of using a trace file, the new NetViz run with NS-2 as it executes. Since the new NetViz did not provide data at start up, the data was streamed from NS-2.

User interface of the new NetViz has controls for starting, pausing and stopping the visualization. It also provides speed control and seeking to a specific time in the visualization. Using these controls users can pause, resume or completely stop the visualization during the execution of NS-2. Currently, among the multiple commands on the command tab in the prototype design, the only command that works on this tab is changing "queue size" This research provided full loop simulation executions including rendering the visualization and providing command feedback to test these dynamic interactions. Start, pause, resume and change queue size commands tested and validated visualizing JSE scenario in Chapter IV. It proved that without multiple iterations or post processing techniques, NetViz users can perform some of purposed tasks. Thus, with the usage of the *mediator*, the static NetViz is taken to the next level bringing the capability of controlling network parameters, as NS-2 executes.

5.1.3 Simulation Execution Scenarios Including Multiple Clients. This research built two simulation execution scenarios including multiple NetViz clients. First execution scenario was included two NetViz clients. Then the number of NetViz clients was increased to six and the second execution scenario was obtained. The NetViz clients run on different operating system were used in this application. This research accomplished the real-time interaction between the elements of these scenarios successfully. Thus, it's proved that multiple NetViz clients can be linked to NS-2 using *mediator* tool even if they work on different operating systems. Additionally, this research obtained the expected results enabling users of NetViz clients to display same simulation and customize the display differently in the direction of purposed tasks.

5.2 Research Limits

A number of modifications had to done in some NS-2 script files in this research because of the limitations of NS-2. The reason for this is that NS-2 does not originally support some of the properties needed to display such visualization functions as clustering or color coding. These modifications were discussed in Chapter III. To integrate the commands discussed in the following section on CCT panel, a modification is needed in NS-2 code for the future work.

Our implementation is also limited by the restrictions of the java swing. Because the Java swing tables are not flexible enough to implement filters settings table, we could not implement this table, as mentioned in Chapter IV. In addition to this, to implement color coding function for the forces, we had to hard-code blue, gray and green colors in the NetViz properties file. The reason for this was that there were force names that need to be matched with the colors to indicate force sides. With the modification in the NS-2 script file, we had the ability to indicate force sides using hard-coded colors in a non-flexible way. The other limit was time to complete command and statistics tab's functionalities. These tabs are discussed in the following section.

5.3 Future Work

The configurable command tool, CCT, including a number of commands was created as a part of [12]. Coupling the CCT panel with NetViz to send commands such as “Turn ON/OFF CBR” , “Change CBR interval”, “Move CBR” , “ Move Wireless Node” successfully. The future purpose is to integrate these commands on the command tab of the new NetViz. But, since NetViz does not have enough information that CCT panel has, integrating these commands requires additional modifications in NS-2 code in future. Thus, command tab functionality could be accomplished by the means of advancing the interaction with NS-2.

Currently, the statistic tab functionality is proceeding to analyze the scenario on the fly. Knowing that graphs are essential in network management and analysis, this research analyzed a simple wired scenario in detail to obtain 2D and 3D graphs using *nans* and *tracegraph*, post processing tools. Usage of these post processing tools provided us a useful foresight which would help the progress of NetViz framework giving a potential to plug these graphs into the user interface. The most important statistics which can be obtained from the trace files are throughput, end to end delay, RTT and the graphs of them. To make NetViz capable of calculating these parameters to analyze and characterize network is a future challenge.

Bibliography

1. Andy Hunt and Thomas Hermann, "The importance of interaction in sonification," Proceedings of ICAD 04, Sydney, Australia, Tech. Rep. 1, July 2004.
2. Kevin Fall and Kannan Varadhan, "The ns manual," UC Berkeley, The VINT Project, January 2009, http://www.isi.edu/nsnam/ns.pdf, Site accessed January 5, 2009.
3. D. Estrin, M. Handley, J. Heidemann, S. McCanne, Y. Xu, and H. Yu., "Network visualization with the vint network animator nam." USC Computer Science Department, Technical Report 99-703b, October 1999.
4. M. Greis, "Tutorial for the network simulator (ns)," VINT group, URL www.isi.edu/nsnam/ns/tutorial, Site accessed January 5, 2009.
5. J. Mark Belue, Captain, USAF, *Network Visualization Design Using Prefuse Visualization Toolkit.* Master's thesis, Air Force Institute of Technology, March 2008.
6. Theodor Holm Nelson, *The Right Way to Think About Software Design*, the art of human computer interface design ed. Reading, MA: Addison-Wesley Publishing Co, 1990.
7. John M. Lewis, Captain, USAF, *Requirements, Design and Prototype of Virtual User Interface for the AFIT Virtual Spaceplane.* Master's thesis, Air Force Institute of Technology, December 1997.
8. Jeffrey Heer , "Prefuse," January 2009, URL http://www.prefuse.org, Site accessed January 5, 2009.
9. Jeffrey Heer, "Prefuse," October 2008, URL http://www.infovis-wiki.net/index.php/Prefuse, Site accessed January 5, 2009.
10. Jaroslaw Malek, "Tracegraph," October 2007, URL http://www.tracegraph.com/, Site accessed January 5, 2009.
11. Ankur Jain, "Nans," URL http://www.geocities.com/ankurjain009/projects.htm, Site accessed January 5, 2009.
12. John S. Weir, Major, USAF, *Mediated User-Simulator Interactive Command with Visualization (MUSIC-V).* Master's thesis, Air Force Institute of Technology, March 2009.
13. Aaron Kershenbaum, Keitha Murray, "Visualization of network structures," Consortium for Computing Sciences in Colleges, December 2005).
14. Enrique Campos-Nanez, "nscript user manual," Department of System Engineering University of Virginia, March, 2001, http://home.gwu.edu/ ecamposn/nscript/nss.pdf, Site accessed January 5, 2009.

15. "Model-view-controller (mvc)," eNode company, Markup Language, 2003, http://www.enode.com/x/markup/tutorial/mvc.html, Site accessed January 5, 2009.

16. Kotilainen Niko, Vapa Mikko, Auvinen Annemari, Weber Matthieu, Vuori Jarkko , "P2pstudio - monitoring, controlling and visualization tool for peer-to-peer networks research," Department of Mathematical Information Technology University of Jyvskyl, Finland, Tech. Rep., October 2006.

17. A. Auvinen, M. Vapa, M. Weber, N. Kotilainen, and J. Vuori , "Chedar: Peer-to-peer middleware," Proceedings of the 20th IEEE International Parallel and Distributed Processing Symposium,Rhodes Island, Greece, 2006.

18. Annemari Auvinen, "Chedar p2p platform," Department of Mathematical Information Technology University of Jyvskyl, Finland, 2003, http://tisu.it.jyu.fi/cheesefactory, Site accessed January 5, 2009.

19. M. Vapa, N. Kotilainen, A. Auvinen, H. Kainulainen, and J. Vuori, "Resource discovery in p2p networks using evolutionary neural networks," International Conference on Advances in Intelligent Systems - Theory and Applications, Luxembourg, 2004.

20. Robert Ball, Glenn A. Fink, Chris North, "Home centric visualization of network traffic for security administration," Department of Computer Science Virginia Polytechnic Institute and State University, Blacksburg, Virginia 2004.

21. Jeffrey Heer, Danah Boyd, "Vizster: Visualizing online social networks," Computer Science Division University of California, Berkeley, 2004.

22. Prashant Rajvaidya, Kevin C. Almeroth, K Claky, "A scalable architecture for monitoring and visualizing multicast statistics," Department of Computer Science University of California, Santa Barbara April 2000.

23. B. Hukaker, E. Nemeth, and K. Claky, "Otter: A general-purpose network visualization tool," in INET, (San Jose, California, USA), June 1999.

24. R. Periakaruppan, "Geoplot - a general purpose geographical visualization tool." http://www.caida.org/Tools/GeoPlot/.

25. T. Bates, R. Chandra, D. Katz, and Y. Rekhter, "Multipro-tocol extensions for bgp-4," Internet Engineering Task Force (IETF), RFC 2283, February 1998.

26. Catherine M. Burns, Johnson Kuo, Sylvia Ng, "Ecological interface design: a new approach for visualizing network management," Advanced Interface Design Lab, Department of Systems Design Engineering, Canada, February 2003.

27. Takayuki Itoh, Hiroki Takakura, Atsushi Sawada, Koji Koyamada, "Hierarchical visualization of network intrusion detection data," Published by the IEEE Computer Society, April 2006.

28. Judith E.Terrill, "Scientific visualization," ITL MCSD Scientific Application and Visualization Group, http://www.math.nist.gov/mcsd/savg/vis/index.html, Site accessed January 5, 2009.

29. Andy Hunt, Thomas Hermann, "The importance of interaction in sonification," Proceedings of ICAD 04-Tenth Meeting of the International Conference on Auditory Display, Sydney, Australia, July, 2004.

30. Matthias Rauterberg, *Usability Engineering.* Swiss Federal Institute of Technology Zrich, 1996.

31. Ivan Herman, Guy Melanon, M. Scott Marshall, "Graph visualization and navigation in information visualization: a survey," Centre for Mathematics and Computer Sciences (CWI), Amsterdam, The Netherlands, December 2005).

32. D. Kimelman, B. Leban, T. Roth, and D. Zernik, "Reduction of visual complexity in dynamic graphs," Proceedings of the Symposium on Graph Drawing 93, Springer-Verlag, 1994.

33. P. Eades and Q.-W. Feng, "Multilevel visualization of clustered graphs," Proceedings of the Symposium on Graph Drawing GD '96, Springer-Verlag, pp. 101-112, 1997.

34. G.J. Wills, "Niche works - interactive visualization of very large graphs," Proceedings of the Symposium on Graph Drawing GD '97, Springer-Verlag, pp. 403-415, 1998.

35. Marcus, Aaron, "Designing graphical user interfaces: Part ii." UnixWorld, Vol. 7, No 10, October 1990, pp. 135-138.

36. Schneiderman Ben, Plaisant Catherine , *Designing The User Interface*, 4th ed. Reading, MA: Pearson Education, 2004.

37. Aliff Umair Salleh, Zulkifli Ishak , Norashidah Md. Din, Md Zaini Jamaludin, "Trace analyzer for ns-2," 4th Student Conference on Research and Development, Shah Alam, Selangor, MALAYSIA, June, 2006.

38. Michael Karl, "A comparison of the architecture of network simulators ns-2 and tossim," Performance Simulation of Algorithms and Protocols, Germany, January 2005.

39. Jae Chung and Mark Claypool, "Ns by example," Worcester Polttechnic Institute, Computer Science.

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)* 26–03–2009	**2. REPORT TYPE** Master's Thesis	**3. DATES COVERED** *(From — To)* May 2007 — Mar 2009

4. TITLE AND SUBTITLE DYNAMIC INTERACTIONS FOR NETWORK VISUALIZATION AND SIMULATION	**5a. CONTRACT NUMBER** DACA99–99–C–9999
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Çiğdem Yetişti, First Lieutenant, TUAF	**5d. PROJECT NUMBER** ENG-09-200
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology, Graduate School of Engineering and Management (AFIT/EN), 2950 Hobson Way, WPAFB OH 45433-7765	**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT/GE/ENG/09-50
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) AFOSR, Complex Network Branch, Mr. Robert Bonneau, Phone Number: (703) 696-9545, rbonneau@gmail.com, 875 N. Randolph, Ste.325, Rm. 3112, Arlington Virginia, 22203	**10. SPONSOR/MONITOR'S ACRONYM(S)** AFOSR/CN
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approval for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This research effort examines related interface studies, interactive techniques and visualization functions to present the design, development and implementation of a new network visualization framework design. From the interface design perspective, this research presents a prototype design to ease the implementation process. The prototype design is introduced to attempt to better facilitate using multiple network visualization functions meeting user needs. With the usage of the mediator tool, the new network visualization framework design interacts with NS-2.The visualization functions such as clustering, labeling, color coding and selecting nodes help accessing network objects and events, supporting four tabs consisting of buttons, menus, and sliders. Interactivity and visualization functions empower the framework to handle the inherent complexity of large networks, allowing the user to interact with the current display of the framework and control the network through the visualization.

15. SUBJECT TERMS

Network, Visualization, Network Simulator, Network Visualization, Visualization Functions, Mediator

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Lt Col Stuart Kurkowski, PhD
U	U	U	UU	96	**19b. TELEPHONE NUMBER** *(include area code)* (937) 785–3636, ext 7228 skurkows@afit.edu

Standard Form 298 (Rev. 8–98)
Prescribed by ANSI Std. Z39.18